Engineering Science and Technology, an International Journal 24 (2021) 1415–1427

HOSTED BY

Contents lists available at ScienceDirect

Engineering Science and Technology, an International Journal

journal homepage: www.elsevier.com/locate/jestch

Path planning of a mobile robot among curved obstacles through tangent drawing and trapezoidal decomposition

Neeta A. Eapen [a,b]

[a] National Institute of Technology Calicut, India
[b] University of Idaho, USA

ARTICLE INFO

Article history:
Received 11 November 2020
Revised 23 January 2021
Accepted 2 March 2021
Available online 11 April 2021

Keywords:
Path planning
Mobile robot
Curved obstacle
Tangent

ABSTRACT

Most of the previous studies on mobile robot path planning consider the obstacles as polygons. However, the complex shaped obstacles should be considered as curves rather than polygons, since the latter may result in non-optimal paths. Three new algorithms are proposed for finding the shortest path of a mobile robot among curved obstacles: **TAD** which uses a **ta**ngent **d**rawing method, **TRAD** which uses **tra**pezoidal **d**ecomposition, and **TAD-TRAD** which uses a combination of the TAD and the TRAD algorithms. The time complexities of the TAD algorithm, the TRAD algorithm, and the TAD-TRAD algorithm are $O(n^3 \log n)$, $O(n \log n)$, and $O(n(\log n)^2)$ respectively, where n is the total number of arcs in all the obstacles. The TAD and the TAD-TRAD algorithms compute an optimal shortest path, whereas the TRAD algorithm computes an approximate shortest path, among curved obstacles. The TAD algorithm has a lower time complexity than existing algorithms which are implemented for the computation of the optimal shortest path. The TRAD algorithm has a lower time complexity than existing algorithms which compute an approximate shortest path and the TAD-TRAD algorithm has a lower time complexity than existing algorithms which compute an optimal shortest path. The experimental comparison of the TAD, TRAD, and TAD-TRAD algorithms on 200 randomly generated problems shows that the TAD-TRAD algorithm has the least running time. Moreover, the TAD-TRAD algorithm computes an optimal path using a lesser number of tangents and with lesser time complexity when experimental comparisons are performed with existing algorithms in the literature.

1. Introduction

A mobile robot has many applications since it can transfer goods from one location to another. For example, a mobile robot is used to transfer molten metal from one location to another in a factory environment. The obstacles to be avoided by the mobile robot are often considered as polygons [23,27]. A curved obstacle may be approximated to a polygon by sampling the boundary of the curved obstacle into a set of points. But the accuracy of the path computed among such polygonized obstacles depends on the sampling technique used for the polygonization of the curved obstacles. The obstacles may have complex shapes, and hence polygonizing them may result in non-optimal paths [20,30]. Therefore, the obstacles should be considered as curves rather than polygons [22,21,4,17]. This work aims to find the shortest path between two locations to be followed by a mobile robot among curved obstacles.

Path planning of a mobile robot is a very challenging area of research. The shortest path for a mobile robot among polygonal obstacles has been widely learned (example [11–13,16,25,34]). But very little research has been done to find the shortest path among curved obstacles (example [6,14,26,30]). The shortest path problem among polygonal obstacles (polygonal domain) computes the shortest path between a source point S and an end point E among a set of polygonal obstacles. This path consists of a set of straight lines. On the other hand, the shortest path problem among curved obstacles (curved domain) computes the shortest path between a source point S and an end point E among a set of curved obstacles C, and this path consists of straight lines as well as curved portions of C.

This work proposes three new algorithms to compute the shortest path for a mobile robot between two points among curved obstacles. Each obstacle consists of a set of arcs. The arcs are the curved edges in a curved obstacle. The first algorithm finds the shortest path using a **ta**ngent **d**rawing method and is named as **TAD**. This algorithm finds the optimal path between two points for a mobile robot among curved obstacles. The TAD algorithm

E-mail address: neetaaeapen@gmail.com

https://doi.org/10.1016/j.jestch.2021.03.002
2215-0986/

has a time complexity of $O(n^3 \log n)$, where n is the total number of arcs in all the obstacles. The time complexity of the TAD algorithm is lesser than all existing algorithms which are implemented [30,28] for the computation of the optimal shortest path among curved obstacles. The number of tangents constructed in the TAD algorithm is the same as that of the best existing theoretical algorithm [6] for the computation of the optimal shortest path.

The second algorithm obtains the shortest path using **tra**pezoidal **d**ecomposition and is named as **TRAD**. This algorithm aims at computing the shortest path for a mobile robot in lesser time by allowing approximate results. This algorithm has a time complexity of $O(n \log n)$, where n is the total number of arcs in all the obstacles. No other algorithm has been developed for finding the shortest approximate path among curved obstacles in less than $O(n \log n)$ time. Hence, this algorithm can be used for fast computations of an approximate path for a mobile robot among curved obstacles.

The third algorithm is a combination of the TAD algorithm and the TRAD algorithm and is named as **TAD-TRAD**. The TAD-TRAD algorithm gives an optimal path in less time. This algorithm has a time complexity of $O(n(\log n)^2)$, where n is the total number of arcs in all the obstacles. No other algorithm has been developed for finding the shortest optimal path among curved obstacles in less than $O(n(\log n)^2)$ time. Hence, this algorithm can be used for fast computations of an optimal path for a mobile robot among curved obstacles.

The TAD algorithm, the TRAD algorithm, and the TAD-TRAD algorithm are applied to 206 input problems, of which 200 problems are randomly generated, and 6 problems are taken from the literature. The comparison of the TAD algorithm, the TRAD algorithm, and the TAD-TRAD algorithm to existing works in literature is made based on the time complexity, nature of the path computed (optimal or approximate), and the support for non-convex obstacles.

This paper is organized as follows. Section 2 gives a literature review of relevant works. Section 3 gives an illustration of the extraction of boundaries of obstacles from a real life situation. Section 4 explains the definitions used in this work. Section 5, Section 6, and Section 7 explain the method and time complexity of the TAD algorithm, the TRAD algorithm, and the TAD-TRAD algorithm, respectively. Section 8 consists of two parts: Section 8.1 gives the experimental comparison of the TAD algorithm, the TRAD algorithm, and the TAD-TRAD algorithm using 200 randomly generated problems. Section 8.2 compares the TAD algorithm, the TRAD algorithm, and the TAD-TRAD algorithm with the existing works in literature. Section 9 gives the conclusion of this paper.

2. Literature review

Kapoor and Maheshwari [15] developed an algorithm for shortest path in the presence of m polygonal obstacles with a time complexity of $O(m^2 \log n + n \log n)$, where n is the total number of points in the obstacles. Ghosh and Mount [12] developed an algorithm for computing visibility graphs in a polygonal domain with a time complexity of $O(n \log n + E)$, where $E = O(n^2)$ is the number of edges in the visiblity graph and n is the total number of vertices in the obstacles. Storer and Reif [34] computed the shortest path in a polygonal domain in $O(kn)$ time, where k is the number of connected components in the obstacle space and n is the size of the obstacle space. Mitchell [25] developed an algorithm for shortest path in the presence of polygonal obstacles in $O(n^{5/3+\epsilon})$ time, where $\epsilon > 0$ is a constant and n is the total number of vertices in the obstacles. Kapoor et al. [16] computed the shortest path in a polygon containing h holes and a total of n vertices with a time complexity of $O(n + h^2 \log n)$. Hershberger and Suri [13] developed an algorithm for finding the shortest path among polygonal obstacles with a time complexity of $O(n \log n)$, where n is the total number of vertices in the obstacle polygons.

Song, Wang, and Zou [33] used an improved particle swarm optimization (PSO) for the path planning of a mobile robot among polygonal obstacles. Sarkar, Barman, and Chowdhury [31] used domain knowledge based genetic algorithm for the path planning of a mobile robot among polygonal obstacles. Wang and Meng [35] used generalized Voronoi graph and multiple potential functions for the path planning of the mobile robot using polygonal obstacles. Li et al. [18] used a combination of potential functions based RRT$*$ and quick RRT$*$ for the path planning of mobile robots among polygonal obstacles.

Very few works have been done to find the shortest path between two points among curved obstacles. The shortest path in curves can be computed from visibility graphs [8] by performing Dijkstra's shortest path algorithm [7] on the visibility graph. Pocchiola and Vegter [26] introduced an algorithm for visibility graph construction of n convex curves using pseudo-triangulations in $O(n \log n + k)$ time where $k = O(n^2)$. Shortest optimal path computation on the constructed visibility graph takes $O(n^2 \log n)$ time. Therefore, the total time complexity for the shortest path computation using this method is $O(n^2 \log n + k)$. This algorithm is developed for only convex curves.

Chen and Wang [5] computed the shortest optimal path among n convex objects between two points s and t in $O(n \log n + k)$ time where $k = O(n^2)$. In this work, after the construction of visibility graphs, a coalesced graph is constructed, which reduces the number of vertices from $O(n^2)$ to $O(n)$. Shortest path computation on the coalesced graph takes $O(n \log n)$ time. This results in a total time complexity of $O(n \log n + k)$ for finding the shortest path. This algorithm is developed for only convex curves.

Chen and Wang [6] calculated the shortest optimal path between two points among h splinegons, with a total of n arcs, in $O(n \log n + k)$ time or $O(n + h\log^{1+\epsilon} h + k)$ time where $\epsilon > 0$ is a small constant and $k = O(h^2)$. In this work, the shortest paths in a splinegon domain (SPSD) is reduced to an instance of the convex SPSD and then solved.

Bharath and Ramanathan [28,30] presented an algorithm for finding the shortest optimal path between two points S and E, among h inner curved obstacles surrounded by a closed outer curve with a total of n arcs, by a tangent removal method using a region elimination procedure. The total number of tangents is $T = 4(nC_2 + nh + n + h)$. An improvement of worst-case running time in [30] from $O(n^4)$ to $O(n^3)$ is possible [29,36] by using the method of Chen and Wang [6].

Hershberger, Suri, and Yildiz [14] developed a near-optimal algorithm for shortest path between two points among curved obstacles, with a total of n arcs, in $O(n \log n + n \log 1/\epsilon)$, where ϵ is the relative error. This work is done using a continuous Dijkstra approach which reduced the time complexity, but it provides an approximate shortest path.

Ajeil et al. [3] used hybridized Particle Swarm Optimization-Modified Frequency Bat (Hybrid PSO-MFB) algorithm for the path planning of a mobile robot. This work is developed for circular obstacles. The robot path planning problem is NP hard and hence [3] has an exponential time complexity.

In summary, out of the few works which compute the shortest path among curved obstacles, very few works support both convex and non-convex curves. Among these, [6] computes the shortest optimal path with the least time complexity and [14] computes

the shortest approximate path with the least time complexity. But [6] computes the shortest optimal path in quadratic time $(O(n \log n + h^2))$ and [14] takes more than linearithmic time $(O(n \log n + n \log 1/\epsilon))$ to compute the shortest approximate path.

2.1. Contributions of this work

This work proposes three new algorithms, TAD, TRAD, and TAD-TRAD, to compute the shortest path for a mobile robot among curved obstacles. These algorithms can be used to find the shortest path among convex and non-convex curved obstacles. The TAD algorithm, the TRAD algorithm, and the TAD-TRAD algorithm has a time complexity of $O(n^3 \log n), O(n \log n)$, and $O(n(\log n)^2)$ respectively, where n is the total number of arcs in all the obstacles. The TAD algorithm and the TAD-TRAD algorithm computes an optimal path for a mobile robot among curved obstacles, whereas the TRAD algorithm computes an approximate path for a mobile robot among curved obstacles. The major contributions of this work are:

- The TAD algorithm has a lower time complexity than exiting algorithms which are implemented [30,28] for the optimal shortest path computation between two points among curved obstacles. The number of tangents constructed in the TAD algorithm is the same as the number of tangents constructed in the best existing theoretical algorithm [6].
- The TRAD algorithm has a lower time complexity than all the existing algorithms which compute the shortest approximate path between two points among curved obstacles.
- The TAD-TRAD algorithm has a lower time complexity than all the existing algorithms which compute the shortest optimal path between two points among curved obstacles. The number of tangents constructed in the TAD-TRAD algorithm is lower than the number of tangents constructed in all the existing algorithms which compute the shortest optimal path between two points among curved obstacles.

3. Illustration of the extraction of boundaries of obstacles from a real life situation

The algorithms proposed in this work computes the shortest path of a mobile robot among curved obstacles in 2-dimension. The 2-dimensional view of real life situations can be extracted for the purpose of shortest path computation. Fig. 1 shows an illustration of the extraction of boundaries of obstacles from a real life situation. Fig. 1a shows a 3-dimensional view of a room containing a table, four chairs, and a cupboard (closet). Imagine a situation where a mobile robot has to move through the room. The table, chairs, and the cupboard are considered as obstacles in this situation. Fig. 1b shows the 2-dimensional view of the boundaries of the obstacles in Fig. 1a. The 2-dimensional view can be used for the computation of shortest path of the mobile robot between two points in the room.

4. Definitions

4.1. Polygonal obstacle

A polygonal obstacle is defined as an obstacle whose boundary consist of only straight lines. For example, the boundary of the cupboard shown in Fig. 1b is a rectangle, and is hence a polygonal obstacle.

4.2. Curved obstacle

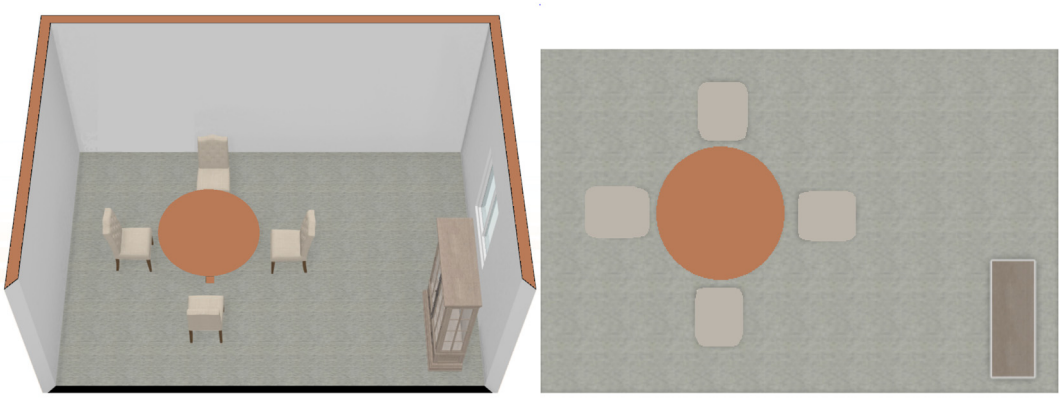
A curved obstacle is an obstacle whose boundary may contain curved lines, and not just straight lines. For example, the boundary of the table shown in Fig. 1b is a circle. Therefore, the table is a curved obstacle. The boundary of the chairs in Fig. 1b consist of both curved lines and straight lines, and is therefore a curved obstacle.

4.3. Tangent

Let A and B be two curved obstacles. Let t be a line segment connecting a point on A and a point on B. t is a tangent to A and B if t is on a locally shortest path [22] (A locally shortest path is defined by [22] as a path which is shortest in the local sense). For example, the red lines in Fig. 2d are tangents.

5. The TAD algorithm

Given a set of curved obstacles C, a source point S, and an end point E, the TAD algorithm aims to find the shortest optimal path for a mobile robot to move from the point S to the point E among the set of curved obstacles C. Each obstacle in C consists of a set of arcs. The arcs are the curved edges in a curved obstacle. S and E are considered as arcs (A point is an arc with zero length). Let A denote the set of arcs (arcs of obstacles, S, and E). Let the total number of arcs in A be n. Note that all the distances mentioned in this paper are Euclidean.



(a) Real life situation (b) Boundaries of obstacles extracted

Fig. 1. Extraction of boundaries of obstacles from a real life situation.

5.1. Method

A **line of vision**, denoted as *LoV*, from a point a to a point b is a tangent t from arc(a) in the direction of b, where arc(a) is an arc containing the point a. If no tangent exists from arc(a) towards b, *LoV* is the straight line from a towards b. Note that when $a = S$ and $b = E$, *LoV* is the straight line from a towards b. The tangent t stops when it hits an arc $A_i \in A$. This arc A_i is the **prospective arc** for the *LoV* from a towards b. If no such arc A_i exists for the *LoV* from a towards b, t stops at b. Then, the point b is visible from the point a, and vice versa. The **visibility test** of an *LoV* from the point a to the point b is performed by verifying whether a prospective arc exists for the *LoV* or not. If a prospective arc exists, the visibility test fails and returns false. Otherwise, it returns true. The **prospective tangents** for the *LoV* from a to b are the common tangents from arc(a) to the prospective arc A_i. There are at most four common tangents between two arcs [2]. Hence, there are at most four prospective tangents between arc(a) and A_i for the *LoV* from a to b. Let t be a prospective tangent for the *LoV* from a to b. Let t_{start} and t_{end} be the start point and end point of t. Let **BD(t)** or **BD** denote the portion of the arc arc(a) from the point a to the point t_{start}. Let **dist(a)** denote the length/distance of the shortest path from S to the point a. Let **P(a)** denote the shortest path from S to the point a.

The algorithm starts by setting $dist(S)$ to zero and $P(S)$ to $[S]$. The point S is marked as seen. Initially, an *LoV* l from the start point S to the end point E is considered. If the visibility test for l returns true, the shortest path from S to E is l and the algorithm stops by setting $dist(E)$ to the length/ distance of l and $P(E)$ to $[S, l]$. Otherwise, the prospective arc for l is obtained. Let $A_i \in A$ be the prospective arc for l. Let T be the set of prospective tangents of l. Let t be a tangent in T. Let t_{start} and t_{end} be the start point and end point of t. For each tangent $t \in T$, the visibility test is conducted.

- Case 1: If the visibility test for t returns true, $dist(t_{end})$ and $P(t_{end})$ is updated as follows. Let $d = dist(t_{start}) + \text{length of } BD(t) + \text{length of } t$
  - If the point t_{end} is already marked,

    $$dist(t_{end}) = \min\{d, dist(t_{end})\}$$

    If $dist(t_{end})$ is updated, $P(t_{end})$ is also updated as

    $$P(t_{end}) = P(t_{start}) \cup BD(t) \cup t$$

  - Otherwise, $dist(t_{end}) = d$, and

    $$P(t_{end}) = P(t_{start}) \cup BD(t) \cup t$$

    The point t_{end} is newly marked as seen.

  If $t_{end} \neq E$, a new *LoV* is considered from arc(t_{end}) to E.
- Case 2: If the visibility test for t returns false, t is considered as an *LoV*.This procedure of finding prospective arcs and prospective tangents for the *LoV*s continues until no more *LoV*s exists. In order to avoid considering an *LoV* more than once, each *LoV* is marked as seen when it is formed. When the TAD algorithm terminates, $P(E)$ contains the shortest path from S to E, and $dist(E)$ contains the length/distance of $P(E)$. $P(E)$ is the path for the mobile robot to move from S to E among the curved obstacles.

The visibility test for an *LoV* is done by verifying if the *LoV* intersects any arc $A_i \in A$. As in [6,9,10,24], the intersection of a line and an arc takes constant time. Since the number of arcs in A is n, visibility test for an *LoV* is performed in $O(n)$ time. Out of all the arcs which intersect with the *LoV*, the prospective arc is the arc closest to the start point of the *LoV*. Therefore, the prospective arc of an *LoV* is determined in $O(n \log n)$ time.

There are at most four common tangents between two arcs [2]. Since the number of arcs in A is $O(n)$, there are at most $O(n^2)$ pair of arcs. Hence, there are at most $O(n^2)$ prospective tangents between the arcs of A. Therefore, at most $O(n^2)$ *LoV*s are constructed in the TAD algorithm.

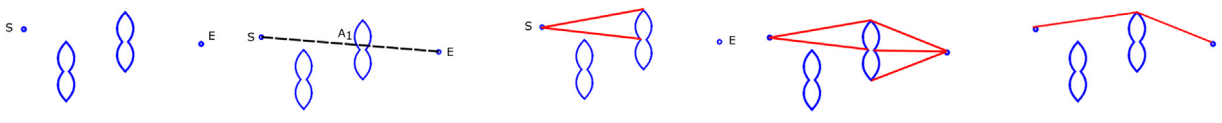
The prospective arc of an *LoV* is determined in $O(n \log n)$ time. The number of *LoV*s which are constructed in the TAD algorithm is at most $O(n^2)$. Therefore, the time complexity for determining the prospective arcs of all the *LoV*s is $O(n^3 \log n)$. Hence, the total time complexity of the TAD algorithm is $O(n^3 \log n)$.

Algorithm 1 provides the pseudocode of the TAD algorithm. The functions which are invoked in Algorithm 1 are present in Algorithm 2.

Fig. 2 shows an illustration of the TAD algorithm. Fig. 2a shows the source point S, the end point E and the curved obstacles. Fig. 2b shows an *LoV* from S to E. The prospective arc is obtained by finding the intersection of all the arcs with the *LoV*. Out of all the arcs which intersect with the *LoV*, A_1 is the closest arc to S. Therefore, the prospective arc for the *LoV* is A_1. Fig. 2c shows the prospective tangents drawn from S to A_1. The visibility test is true for both the tangents. Therefore, the shortest path and distance from S to the tangent end points are updated. New *LoV*s are constructed from the tangent end points, and this procedure continues until no more *LoV*s exists. Fig. 2d shows the tangents constructed in the TAD algorithm. Fig. 2e shows the shortest path obtained.

Algorithm 1: The TAD algorithm

```
   input : A set of arcs A, a source point S, and an
           end point E
   output: P(E) and dist(E)
 1 dist(S) ← 0;
 2 P(S) ← [S];
 3 Mark S as seen;
 4 l ← LoV from S to E;
 5 Initialize a stack X as empty;
 6 Push l to X;
 7 while X is not empty do
 8     l ← pop from X;
 9     Mark l;
10     if visibilityTestForTAD(l) = true then
11         d ←
             dist(l_start) + length of BD(l) + length of l;
12         if l_end is already marked then
13             if d < dist(l_end) then
14                 dist(l_end) ← d;
15                 P(l_end) ← P(l_start) ∪ BD(l) ∪ l;
16             end
17         else
18             dist(l_end) ← d;
19             P(l_end) ← P(l_start) ∪ BD(l) ∪ l;
20             Mark l_end as seen;
21         end
22         if l_end ≠ E then
23             l ← LoV from arc(l_end) to E;
24             Push l to X if it is not marked;
25         end
26     else
27         A_i ← prospectiveArcForTAD(l);
28         T ← common tangents between l_start and
             A_i;
29         foreach t ∈ T do
30             Push t to X if it is not marked;
31         end
32     end
33 end
```



(a) Input (b) LoV (c) Tangents (d) All tangents (e) Shortest Path

Fig. 2. Illustration of the TAD algorithm.

Algorithm 2: Functions for TAD

```
1  Function visibilityTestForTAD(l)
2      foreach A_i ∈ A do
3          if A_i intersects l then
4              Return false;
5          end
6      end
7      Return true;
8  end
9  Function prospectiveArcForTAD(l)
10     Initialize a set Y to empty;
11     foreach A_i ∈ A do
12         if A_i intersects l then
13             Add A_i to Y;
14         end
15     end
16     Sort the arcs in Y in the increasing order of
         distance from l_start;
17     Return the first arc in Y;
18 end
```

5.2. Time complexity of the TAD algorithm

The TAD algorithm computes the shortest path between two points among curved obstacles by determining the prospective arcs and by constructing the prospective tangents.

Algorithm 2 contains the functions *visibilityTestForTAD* and *prospectiveArcForTAD*. Since the number of arcs in A is n, line 2 of the function *visibilityTestForTAD* runs at most $O(n)$ times. All lines except line 2 of *visibilityTestForTAD* take only $O(1)$ time. Therefore, time complexity of *visibilityTestForTAD* is $O(n)$.

Since the number of arcs in A is n, line 11 of the function *prospectiveArcForTAD* runs at most $O(n)$ times. Line 16 takes $O(n \log n)$ since it involves the sorting of n arcs. All lines except line 11 and line 16 of *prospectiveArcForTAD* take only $O(1)$ time. Therefore, time complexity of *prospectiveArcForTAD* is $O(n + n \log n) = O(n \log n)$.

The time complexity of Algorithm 1 depends on the number of times the while loop is run. The while loop runs at most $O(n^2)$ times since at most $O(n^2)$ *LoV*s are constructed. Therefore, the functions *visibilityTestForTAD* and *prospectiveArcForTAD* in lines 10 and 27 respectively (in Algorithm 1) is invoked at most $O(n^2)$ times. All lines except lines 7, 10, and 27 take only $O(1)$ time. Therefore, the total time complexity of the TAD algorithm is $O(n^3) + O(n^3 \log n) = O(n^3 \log n)$.

5.2.1. Time complexity in terms of the number of arcs and the number of obstacles

Let h be the number of obstacles. If all the obstacles are convex, the number of tangents (or number of *LoV*s constructed) is $O(h^2)$. When the obstacles are non-convex, the tangents in the corridors [6] should also be considered, in addition to the $O(h^2)$ tangents (The explanation of corridors is present in [6]). There can be at most $O(h)$ corridors [6]. Since each corridor has a constant number of tangents, the maximum number of tangents in the corridors is $O(h)$ [6]. Therefore, the number of tangents constructed in the TAD algorithm is $O(h^2) + O(h) = O(h^2)$. Since a prospective arc is determined in $O(n \log n)$ time, the total time complexity of the TAD algorithm in terms of n and h is $O(h^2 n \log n)$.

Therefore the time complexity of the TAD algorithm is:

$$\min\{O(n^3 \log n), O(h^2 n \log n)\}$$

6. The TRAD algorithm

Given a set of curved obstacles C, a source point S, and an end point E, the TRAD algorithm aims to find the shortest approximate path for a mobile robot to move from the point S to the point E among the set of curved obstacles C. Let A denote the set of arcs (similar to TAD). Let the total number of arcs in A be n.

6.1. Method

Section 6.1.1 and Section 6.1.2 are the steps to find the shortest path for a mobile robot, between the source point S and the end point E, among curved obstacles, using the TRAD algorithm.

6.1.1. Trapezoidal decomposition of the arcs

Trapezoidal decomposition of the arcs in A is performed in $O(n \log n)$ time [8]. This divides the space into $O(n)$ trapezoidal cells. Let T denote the set of trapezoidal cells. The arcs in A are associated with the trapezoidal cells in T. Each trapezoidal cell has at most four edges. Two trapezoidal cells are adjacent if they share an edge. Let tc_i denote a trapezoidal cell in T. tc_i is adjacent to at most four trapezoidal cells tc_j, tc_k, tc_l, and tc_m. For each trapezoidal cell $tc_i \in T$, tangents are constructed from the arcs in tc_i to the arcs in tc_j, tc_k, tc_l, and tc_m. There are at most four common tangents between two arcs [2]. Since there are $O(n)$ trapezoidal cells in T, at most $O(n)$ tangents are constructed. The tangents which are constructed is added to a tangent visibility graph $G = (V, E)$. The end points of the tangents are the vertices in G.

Let P_i be the set of tangent end points on an arc A_i. Let BD_{edge} (**boundary edge**) be an edge between the points $a \in P_i$ and $b \in P_i$ with length/distance equal to the length/distance of the portion of the arc A_i between the points a and b. Let $BD_{edgeSet}$ (**boundary edge set**) denote the set of all possible boundary edges from the set of points P_i. The edges in the set $BD_{edgeSet}$ is added to the graph G. The boundary edge set of all the arcs in A is obtained and the edges in each of the boundary edge set are added to the graph G.

There are at most four common tangents between two arcs [2]. Since a trapezoidal cell is adjacent to at most four trapezoidal cells, at most $O(1)$ tangents are incident on an arc A_i. Therefore, the number of tangent end points on an arc is at most $O(1)$. Hence, the number of edges in the boundary edge set of A_i is $O(1)$. Since the number of arcs in A is n, at most $O(n)$ boundary edges are con-

structed and added to the graph G. Hence, the graph G has at most $O(n)$ edges and at most $O(n)$ vertices.

6.1.2. Shortest path from the tangent visibility graph

The shortest path from the source point S to the end point E in the tangent visibility graph G is computed using Dijkstra's shortest path algorithm [7] in $O(|E| + |V| \log |V|) = O(n + n \log n) = O(n \log n)$ time. The computed path is the path for the mobile robot to move from S to E among the curved obstacles.

The pseudocode of the TRAD algorithm is given in Algorithm 3.

Fig. 3 shows an illustration of the TRAD algorithm. Fig. 3a shows the source point S, the end point E and the curved obstacles. Fig. 3b shows the trapezoidal cells obtained from trapezoidal decomposition of the input. The trapezoidal cell T_0 is adjacent to T_1. Therefore, tangents are drawn between S and the arcs of T_1 as shown in Fig. 3c. Similarly, for each pair of adjacent trapezoidal cells, tangents are drawn between the arcs of the adjacent cells. Fig. 3d shows the tangents constructed in the TRAD algorithm. These tangents are added to a graph. Dijkstra's shortest path algorithm is performed on the graph to obtain the shortest path between S and E, as shown in Fig. 3e.

Algorithm 3: The TRAD algorithm

```
   input : A set of arcs A, a source point S, and an
           end point E
   output: Shortest path between S and E
1  Trapezoidal decomposition of the arcs in A is
     performed;
2  Let T denotes the set of trapezoidal cells;
3  Initialize a set S_tangents to store the tangents;
4  foreach tc_i ∈ T do
5      Initialize a set T_adjacent to store the adjacent
         trapezoidal cells;
6      Find the trapezoidal cells adjacent to tc_i and
         add to T_adjacent;
7      foreach tc_j ∈ T_adjacent do
8          if both tc_i and tc_j contains an arc then
9              Construct tangents between arcs of tc_i
                 and tc_j, and add to S_tangents;
10         end
11     end
12 end
13 Let G be a tangent visibility graph;
14 Add the end points of the tangents in S_tangents as
     vertices in G;
15 Add the tangents in S_tangents as edges to G;
16 foreach A_i ∈ A do
17     P_i ← the set of tangent end points on A_i;
18     foreach a ∈ P_i and b ∈ P_i such that a ≠ b do
19         BD_edge ← an edge between a and b with
             length equal to the length of the portion of
             the arc A_i between a and b;
20         Add BD_edge to G;
21     end
22 end
23 Find the shortest path from S to E in G using
     Dijkstra's shortest path algorithm;
```

6.2. Time complexity of the TRAD algorithm

The time complexity of the TRAD algorithm can be seen from Algorithm 3. Line 1 is done in $O(n \log n)$ time [8], where n is the total number of arcs in A. Lines 2 and 3 take only $O(1)$ time. Since there are $O(n)$ trapezoidal cells in T, line 4 runs at most $O(n)$ times. Lines 5, 6, 8, and 9 take only $O(1)$ time. Since a trapezoidal cell has at most four adjacent trapezoidal cells, line 7 runs only $O(1)$ time. Therefore, the time complexity for lines 4–12 is $O(n)$ and the number of tangents in $S_{tangents}$ is at most $O(n)$. Line 13 takes $O(1)$ time. Lines 14 and 15 takes $O(n)$ time since $S_{tangents}$ contains $O(n)$ tangents. Since the number of arcs in A is n, line 16 runs at most n times. Since a trapezoidal cell is adjacent to $O(1)$ trapezoidal cells, the number of points/vertices in P_i of line 17 is $O(1)$. Hence, lines 18–21 has a time complexity of $O(1)$. Therefore, lines 16–22 takes $O(n)$ time. The graph G contains at most $O(n)$ edges and at most $O(n)$ vertices. In line 23, the shortest path from the source point S to the end point E in the graph G is computed using Dijkstra's shortest path algorithm [7] in $O(n \log n)$ time. Hence the total time complexity of the TRAD algorithm is $O(n \log n)$.

6.2.1. Time complexity in terms of the number of arcs and the number of obstacles

Let h be the number of obstacles. If all the obstacles are convex, the number of edges (or number of tangents) in G is $O(h)$. When the obstacles are non-convex, the tangents in the corridors [6] should also be considered, in addition to the $O(h)$ edges (The explanation of corridors is present in [6]). There can be at most $O(h)$ corridors [6]. Since each corridor has a constant number of tangents, the maximum number of tangents in the corridors is $O(h)$ [6]. Therefore, the number of edges in G in terms of h is $O(h) + O(h) = O(h)$. Then, the Dijkstra's shortest path algorithm is performed in $O(h \log h)$ time. Since the number of obstacles can be at most the number of arcs, $h \leqslant n$. Since the trapezoidal decomposition of the arcs is done in $O(n \log n)$ time [8], the time complexity of the TRAD algorithm is $O(n \log n + h \log h) = O(n \log n)$.

6.3. The TRAD algorithm computes an approximate path

The TRAD algorithm computes an approximate path, and not the optimal path, between two points among curved obstacles. The path is approximate because tangents are drawn only between arcs of adjacent cells. Let A_i, A_j, and A_k be three arcs which lie in trapezoidal cells tc_i, tc_j, and tc_k. Assume that tc_i is adjacent to tc_j, tc_k is adjacent to tc_j, and tc_i is not adjacent to tc_k. Let t_1, t_2, and t_3 be tangents from A_i to A_j, from A_j to A_k, and from A_i to A_k, respectively. Let the start and end points of t_1 be t_{1start} and t_{1end}, respectively. Similarly, let the start and end points of t_2 be t_{2start} and t_{2end}, and t_3 be t_{3start} and t_{3end} respectively. Let $t_{1start} = t_{3start}, t_{1end} = t_{2start}$, and $t_{2end} = t_{3end}$ (t_1, t_2, and t_3 can be visualized as the three edges of a triangle). Assume that $[t_3]$ is the shortest optimal path between the points t_{3start} and t_{3end}. The TRAD algorithm constructs tangents between the arcs A_i and A_j, and the arcs A_j, and A_k. The TRAD algorithm does not construct tangents between the arcs A_i and A_k since tc_i is not adjacent to tc_k. Therefore, the path computed by the TRAD algorithm between t_{3start} and t_{3end} is $[t_1, t_2]$. Due to triangle inequality, $(t_1 + t_2) > t_3$. Therefore, the TRAD algorithm computes an approximate path between t_{3start} and t_{3end}.

7. The TAD-TRAD algorithm

Given a set of curved obstacles C, a source point S, and an end point E, the TAD-TRAD algorithm aims to find the shortest optimal path for a mobile robot to move from the point S to the point E among the set of curved obstacles C. Let A denote the set of arcs (similar to TAD and TRAD). Let the total number of arcs in A be n.

7.1. Pre-processing step

Trapezoidal decomposition of the arcs in A is performed in $O(n \log n)$ time [8]. This divides the space into $O(n)$ trapezoidal cells. Let T denote the set of trapezoidal cells. The arcs in A are associated with the trapezoidal cells in T.

7.2. Method

The definitions of *LoV*, prospective arc, prospective tangents and *BD* are already given in Section 5.1. The TAD-TRAD algorithm is performed in almost the same way as in the TAD algorithm. The only difference is in the prospective arc determination and the visibility test, which results in a reduction in time complexity in the TAD-TRAD algorithm.

Let l be an *LoV*. Let l_{start} and l_{end} be the start and the end point of l. The prospective arc for l is determined as follows. Let tc_i be the trapezoidal cell which contains l_{start}. tc_i is adjacent to at most four trapezoidal cells. Find the adjacent trapezoidal cell, tc_j, which intersects with l. tc_j is obtained in $O(1)$ time. As in [6,9,10,24], the intersection of a line and an arc takes constant time. If an arc A_i exists in tc_j, check if A_i intersects with l in $O(1)$ time. If A_i intersects with l, A_i is the prospective arc of l. This procedure of finding the adjacent trapezoidal cell which intersects with l and checking for the existence of a prospective arc in it is repeated until one of the following conditions is satisfied.

- A prospective arc is obtained (Condition 1).
- The trapezoidal cell containing l_{end} is reached (Condition 2).

If the Condition 2 is satisfied, l_{end} is visible from l_{start} and vice versa. The visibility test for an *LoV* is done by verifying if Condition 2 is satisfied or not. The visibility test returns true if the Condition 2 is satisfied. Otherwise, it returns false.

The pseudocode of the TAD-TRAD algorithm is given in Algorithm 4. The functions invoked in Algorithm 4 are given in Algorithm 5.

A query point is located in $O(\log n)$ time from the trapezoidal decomposition of the space, as explained in [8]. The maximum depth between two trapezoidal cells is $O(\log n)$ [8]. Therefore, the prospective arc of an *LoV* is determined by checking for the existence of a prospective arc in at most $O(\log n)$ trapezoidal cells. Since the checking for the existence of a prospective arc in a trapezoidal cell is performed in $O(1)$ time, the prospective arc A_i for an *LoV* is obtained in $O(\log n)$ time. Therefore, the visibility test for an *LoV* is done in $O(\log n)$ time.

*LoV*s are drawn from an arc $A_i \in A$ until the prospective tangents, whose visibility test returns true, are obtained. Since the maximum depth between two trapezoidal cells is $O(\log n)$, at most $O(\log n)$ *LoV*s are drawn from an arc.

There are at most four common tangents between two arcs [2]. At most $O(\log n)$ *LoV*s are drawn from an arc. Since the number of arcs in A is $O(n)$, at most $O(n \log n)$ *LoV*s are constructed in the TAD-TRAD algorithm.

The visibility test for an *LoV* is done in $O(\log n)$ time. Therefore, the prospective arc for an *LoV* is obtained in $O(\log n)$ time. The number of *LoV*s which are constructed in the TAD-TRAD algorithm is at most $O(n \log n)$. Therefore, the time complexity for determining the prospective arcs of all the *LoV*s is $O(n(\log n)^2)$. Hence, the total time complexity of the TAD-TRAD algorithm is $O(n(\log n)^2)$.

Fig. 4 shows an illustration of the TAD-TRAD algorithm. Fig. 4a shows the source point S, the end point E and the curved obstacles. Fig. 4b shows the trapezoidal cells obtained from the pre-processing step (trapezoidal decomposition). Fig. 4c shows an *LoV* from S to E. S is in the trapezoidal cell T_0. T_0 is adjacent to T_1. The *LoV* intersects with T_1, but does not intersect with any arc in T_1. Therefore, the next cell which might contain the prospective arc is considered. T_2 is adjacent to T_1. The *LoV* intersects with T_2. T_2 contains arc A_1 which intersects with the *LoV*. Therefore, the prospective arc for the *LoV* is A_1. Fig. 4d shows the prospective tangents drawn from S to A_1. The visibility test is true for both the tangents. Therefore, the shortest path and distance from S to the tangent end points is updated. New *LoV*s are constructed from the tangent end points and this procedure continues until no more *LoV*s exists. Fig. 4e shows the tangents constructed in the TAD-TRAD algorithm. Fig. 4f shows the shortest path obtained.

Algorithm 4: The TAD-TRAD algorithm

```
input : A set of arcs A, a source point S, and an
        end point E
output: P(E) and dist(E)
Trapezoidal decomposition of the arcs in A is performed;
Let T denotes the set of trapezoidal cells;
dist(S) ← 0;
P(S) ← [S];
Mark S as seen;
l ← LoV from S to E;
Initialize a stack X as empty;
Push l to X;
while X is not empty do
    l ← pop from X;
    Mark l;
    if visibilityTestForTAD − TRAD(l) = true then
        d ← dist(l_start) + length of BD(l) + length of l;
        if l_end is already marked then
            if d < dist(l_end) then
                dist(l_end) ← d;
                P(l_end) ← P(l_start) ∪ BD(l) ∪ l;
            end
        else
            dist(l_end) ← d;
            P(l_end) ← P(l_start) ∪ BD(l) ∪ l;
            Mark l_end as seen;
        end
        if l_end ≠ E then
            l ← LoV from arc(l_end) to E;
            Push l to X if it is not marked;
        end
    else
        A_i ← prospectiveArcForTAD − TRAD(l);
        T_tangents ← common tangents between l_start and A_i;
        foreach t ∈ T_tangents do
            Push t to X if it is not marked;
        end
    end
end
```

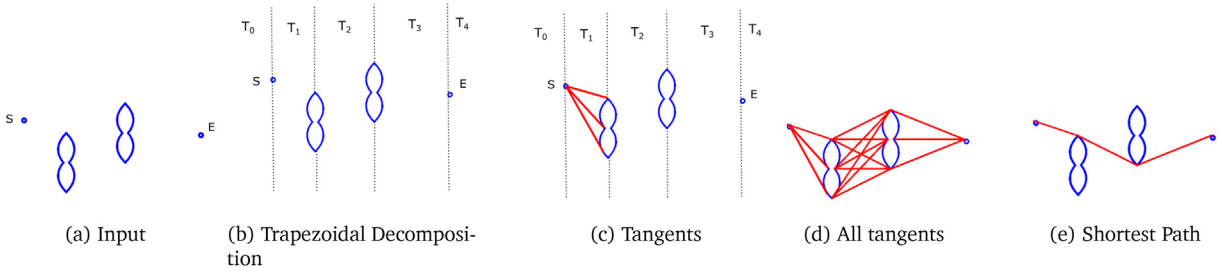


Fig. 3. Illustration of the TRAD algorithm.

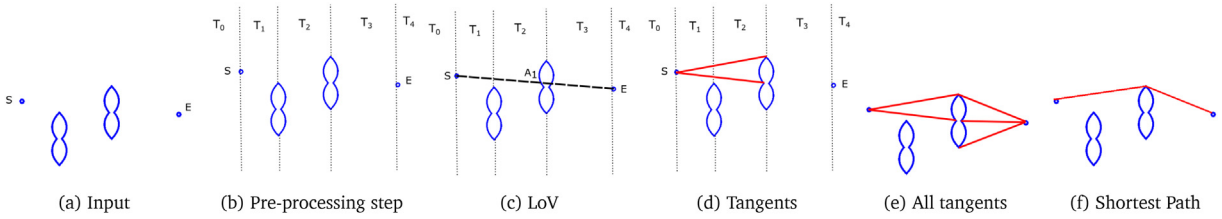


Fig. 4. Illustration of the TAD-TRAD algorithm.

Algorithm 5: Functions for TAD-TRAD

```
1  Function prospectiveArcForTAD-TRAD(l)
2      tc_i ← trapezoidal cell which contains l_start;
3      tc_last ← trapezoidal cell which contains l_end;
4      tc_j ← trapezoidal cell adjacent to tc_i which
         intersects with l;
5      while tc_j ≠ tc_last do
6          if tc_j contains an arc A_i then
7              if A_i intersects with l then
8                  Return A_i;
9              end
10         end
11         tc_i ← tc_j;
12         tc_j ← trapezoidal cell adjacent to tc_i which
             intersects with l;
13     end
14     Return NULL;
15 end
16 Function visibilityTestForTAD-TRAD(l)
17     if prospectiveArcForTAD-TRAD(l)=NULL then
18         Return true;
19     else
20         Return false;
21     end
22 end
```

7.3. Time complexity of the TAD-TRAD algorithm

Other than the prospective arc determination procedure and the visibility test, the shortest path computation of the TAD-TRAD algorithm is similar to the shortest path computation of the TAD algorithm.

Algorithm 5 contains the functions $prospectiveArcForTAD-TRAD$ and $visibilityTestForTAD-TRAD$. It can be seen that the time complexity of the function $prospectiveArcForTAD-TRAD$ depends on the number of times the while loop in line 5 is run. Since the maximum depth between two trapezoidal cells is $O(\log n)$, the while loop in line 5 runs at most $O(\log n)$ time. All lines except line 5 take only $O(1)$ time. Therefore, the time complexity of the function $prospectiveArcForTAD-TRAD$ is $O(\log n)$.

All lines except line 17 of the function $visibilityTestForTAD-TRAD$ take only $O(1)$ time. Clearly, this implies that the time complexity of the function $visibilityTestForTAD-TRAD$ is also $O(\log n)$.

Algorithm 4 gives the TAD-TRAD algorithm. Line 1 is done in $O(n \log n)$ time [8], where n is the total number of arcs in A. The time complexity of Algorithm 4 depends on the number of times the while loop is run. The while loop runs at most $O(n \log n)$ times since at most $O(n \log n)$ *LoV*s are constructed. All lines except lines 1, 9, 12, and 29 take only $O(1)$ time. Since the time complexities of the functions $prospectiveArcForTAD-TRAD$ and $visibilityTestForTAD-TRAD$ are both $O(\log n)$, the time complexity of Algorithm 4 is $O(n(\log n)^2)$.

7.3.1. Time complexity in terms of the number of arcs and the number of obstacles

Let h be the number of obstacles. If all the obstacles are convex, the number of tangents (or number of *LoV*s constructed) is $O(h \log h)$. When the obstacles are non-convex, the tangents in the corridors [6] should also be considered, in addition to the $O(h \log h)$ tangents (The explanation of corridors is present in [6]). There can be at most $O(h)$ corridors [6]. Since each corridor has a constant number of tangents, the maximum number of tangents in the corridors is $O(h)$ [6]. Therefore, the number of tangents constructed in the TAD-TRAD algorithm is $O(h \log h) + O(h) = O(h \log h)$. The pre-processing step is performed in $O(n \log n)$ time. Since a prospective arc is determined in $O(\log n)$ time, the total time complexity of the TAD-TRAD algorithm in terms of n and h is $O(n \log n) + O(h \log h \log n)$.

Therefore the time complexity of the TAD-TRAD algorithm is:

$$\min\{O(n(\log n)^2), O(n \log n + h \log h \log n)\}$$

8. Results and discussions

The algorithms, TAD, TRAD, and TAD-TRAD, are performed on an Intel Core i7-9750H machine with 2.6 GHz clock rate. The TAD algorithm, the TRAD algorithm, and the TAD-TRAD algorithm are applied to 206 input problems, of which 200 problems are ran-

Table 1
Details of the randomly generated input problems.

Number of input problems	200
Minimum number of arcs	44
Median number of arcs	191
Maximum number of arcs	320
Minimum number of obstacles	5
Median number of obstacles	33
Maximum number of obstacles	74

domly generated, and 6 problems are taken from the literature. Section 8.1 describes the performance of the TAD algorithm, the TRAD algorithm, and the TAD-TRAD algorithm for the 200 randomly generated problems. Section 8.2 provides a comparison of this work with existing algorithms in the literature. The comparison of the algorithms in literature with the TAD algorithm, the TRAD algorithm, and the TAD-TRAD algorithm is made based on the time complexities of the algorithms, the nature of the shortest path computed (exact or approximate), and the support for non-convex obstacles. Section 8.2 also provides the experimental comparison of the TAD-TRAD algorithm with [30,3,1,32,19].

8.1. Comparison of the TAD algorithm, the TRAD algorithm, and the TAD-TRAD algorithm through experimental analysis

The TAD algorithm, the TRAD algorithm, and the TAD-TRAD algorithm are applied to 200 randomly generated problems. The details of the randomly generated problems are given in Table 1. Fig. 5 shows a randomly generated input problem and the paths computed using the TAD algorithm, the TRAD algorithm, and the TAD-TRAD algorithm. Fig. 5a shows the input problem, which consists of a set of curved obstacles, the source point *S*, and the end point *E*. Fig. 5b shows the path computed by the TAD algorithm for the input problem shown in Fig. 5a. The TAD algorithm takes 0.055 s to obtain the path shown. The distance/length of the path

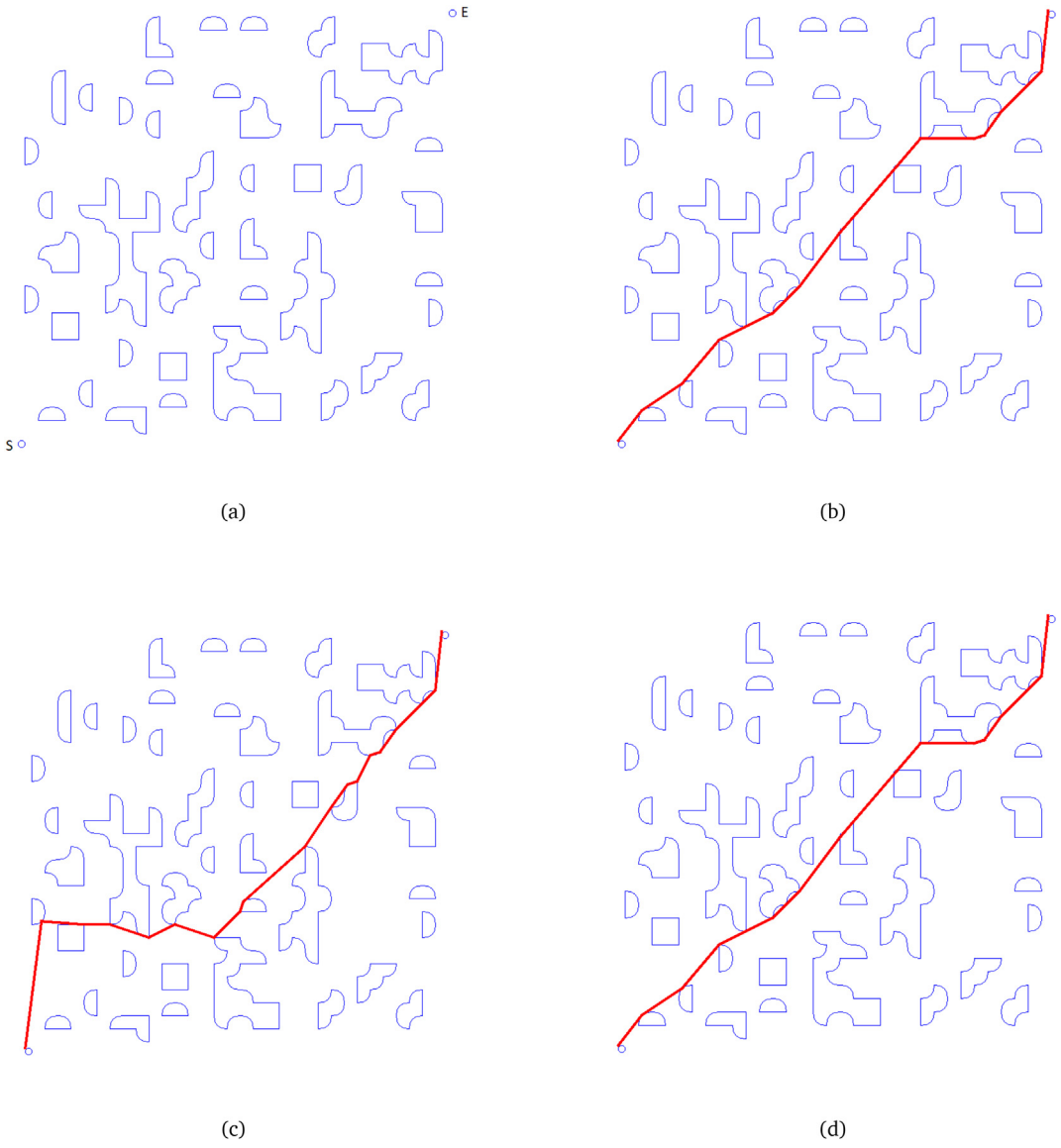


Fig. 5. (a) Input problem consisting of the obstacles, the source point S, and the end point E (b) Path obtained using the TAD algorithm is shown. The distance/length of the path is 253.708 mm. The TAD algorithm takes 0.055 s to obtain the path shown. (c) Path obtained using the TRAD algorithm is shown. The distance/length of the path is 303.487 mm. The TRAD algorithm takes 0.185 s to obtain the path shown. (d) Path obtained using the TAD-TRAD algorithm is shown. The distance/length of the path is 253.708 mm. The TAD-TRAD algorithm takes 0.023 s to obtain the path shown.

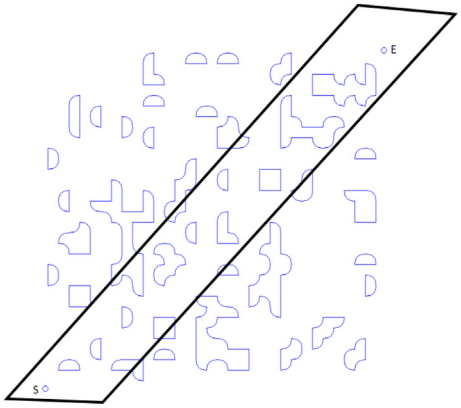


Fig. 6. The tangents are drawn only between the prospective arcs in the quadrilateral shown, for the TAD algorithm and the TAD-TRAD algorithm.

is 253.708 mm. Fig. 5c shows the path computed by the TRAD algorithm for the input problem shown in Fig. 5a. The TRAD algorithm takes 0.185 s to obtain the path shown. The distance/length of the path is 303.487 mm. Fig. 5d shows the path computed by the TAD-TRAD algorithm for the input problem shown in Fig. 5a. The TAD-TRAD algorithm takes 0.023 s to obtain the path shown. The distance/length of the path is 253.708 mm. It can be noted that the path computed by the TAD algorithm and TAD-TRAD algorithm is shorter than the path computed by the TRAD algorithm. This is because the TAD algorithm and TAD-TRAD algorithm computes an optimal path, whereas the TRAD algorithm computes an approximate path. The TAD algorithm, the TRAD algorithm, and the TAD-TRAD algorithm has a time complexity of $O(n^3 \log n), O(n \log n)$, and $O(n(\log n)^2)$ respectively, where n is the total number of arcs. It can also be noted that even though the time complexity of the TRAD algorithm is theoretically smaller than the TAD algorithm and the TAD-TRAD algorithm, the path computation by the TAD algorithm and the TAD-TRAD algorithm takes lesser time than that of the TRAD algorithm. This is because the TRAD algorithm constructs tangents between arcs of all adjacent cells, whereas the TAD algorithm and the TAD-TRAD algorithm constructs tangents only between a subset of the arcs in the input problem. The TAD algorithm and the TAD-TRAD algorithm constructs tangents only between the arcs inside the quadrilateral shown in Fig. 6.

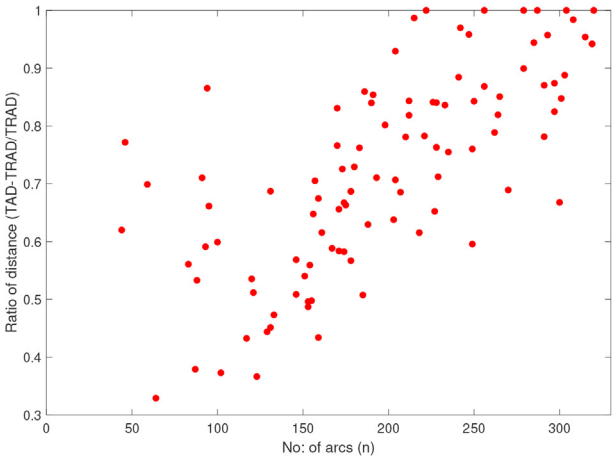


Fig. 7. The x-axis is the number of arcs (n). The y-axis is the ratio of distance, where ratioofdistance $= d_{optimal}/d_{TRAD}, d_{optimal}$ is the length(distance) of the path computed using the TAD-TRAD algorithm, and d_{TRAD} is the length(distance) of the path computed using the TRAD algorithm.

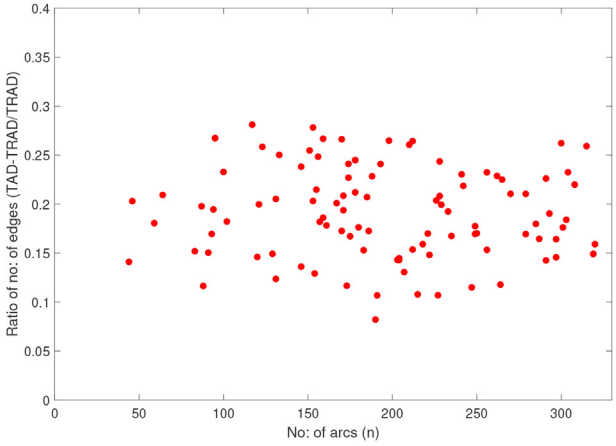


Fig. 8. The x-axis is the number of arcs (n). The y-axis is the ratio of no: of edges, where ratioofno : ofedges $= e_{optimal}/e_{TRAD}, e_{optimal}$ is the number of edges(tangents) constructed for the path computation using the TAD-TRAD algorithm, and e_{TRAD} is the number of edges(tangents) constructed for the path computation using the TRAD algorithm.

Since the TAD algorithm and the TAD-TRAD algorithm computes an optimal path among curved obstacles, the path generated by the TAD algorithm and the TAD-TRAD algorithm is the same. Let $d_{optimal}$ denote the length(distance) of the path computed using the TAD-TRAD algorithm (or the TAD algorithm). Let d_{TRAD} denote the length(distance) of the path computed using the TRAD algorithm. Let **ratioofdistance** $= d_{optimal}/d_{TRAD}$. In Fig. 7, the x-axis and the y-axis show the number of arcs (n) and the ratio of distance respectively. It shows that the ratio of distance is in the range of 0.3 to 1. Therefore, the path computed by the TAD algorithm and the TAD-TRAD algorithm is smaller than the path computed by the TRAD algorithm.

Since the computation of the shortest path using the TAD algorithm and the TAD-TRAD algorithm is through the tangent drawing method, both the TAD algorithm and the TAD-TRAD algorithm generates optimal paths using the same number of tangents. Let $e_{optimal}$ denote the number of edges(tangents) constructed for the path computation using the TAD-TRAD algorithm (or the TAD algorithm). Let e_{TRAD} denote the number of edges(tangents) constructed for the path computation using the TRAD algorithm. Let **ratioofno** : **ofedges** $= e_{optimal}/e_{TRAD}$. In Fig. 8, the x-axis and the y-axis shows the number of arcs (n) and the ratio of no: of edges respectively. It shows that the ratio of no: of edges is in the range of 0.05 to 0.4. Therefore, the number of edges(tangents) constructed for the path computation using the TAD algorithm and the TAD-TRAD algorithm is much lesser than that of the TRAD algorithm.

In Fig. 9, the x-axis and the y-axis are the number of arcs (n) and the number of obstacles respectively. In Fig. 9a, the color bar gives the running time of the TAD algorithm (in seconds) for the 200 randomly generated input problems. It can be seen that the running time has a greater dependence on the number of arcs (n) than the number of obstacles. The running time of the TAD algorithm is in the range 0 to 0.11 s. In Fig. 9b, the color bar gives the running time of the TRAD algorithm (in seconds) for the 200 randomly generated input problems. It can be seen that the running has a greater dependence on the number of arcs (n) than the number of obstacles. The running time of the TRAD algorithm is in the range 0 to 0.45 s. In Fig. 9b, the color bar gives the running time of the TAD-TRAD algorithm (in seconds) for the 200 randomly generated input problems. It can be seen that the running has a greater

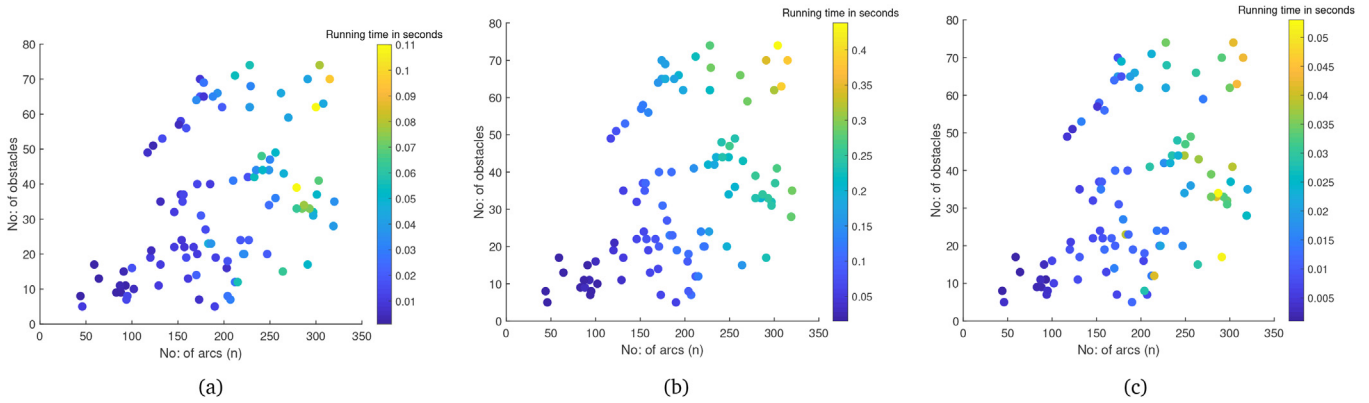


Fig. 9. The x-axis and the y-axis are the number of arcs (n) and the number of obstacles respectively (a) The color bar gives the running time of the TAD algorithm (in seconds) (b) The color bar gives the running time of the TRAD algorithm (in seconds) (c) The color bar gives the running time of the TAD-TRAD algorithm (in seconds).

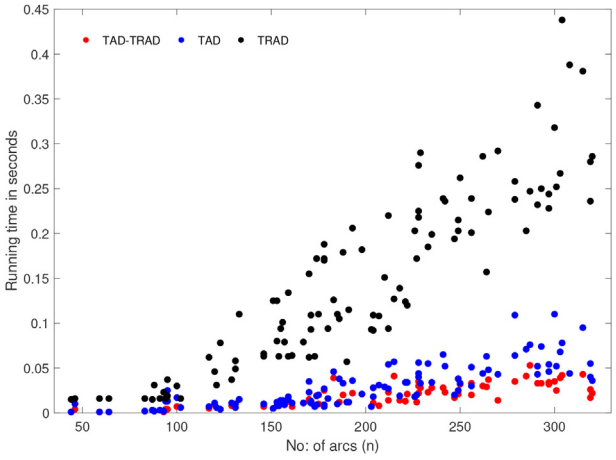


Fig. 10. Comparison of running time (in seconds) of the TAD algorithm, the TRAD algorithm, and the TAD-TRAD algorithm.

dependence on the number of arcs (n) than the number of obstacles. The running time of the TAD-TRAD algorithm is in the range 0 to 0.055 s. Fig. 9 shows that the running time has a greater dependence on the number of arcs than the number of obstacles for all the three algorithms.

Fig. 10 shows the comparison of the running time in seconds for the TAD algorithm, the TRAD algorithm, and the TAD-TRAD algorithm, for the 200 randomly generated input problems. The x-axis and the y-axis show the number of arcs (n) and the running time in seconds, respectively. It can be seen that the TAD-TRAD algorithm has the least running time for all the problems. It can also be noted that the TAD algorithm has a lower running time than the TRAD algorithm.

8.2. Comparison with algorithms in the literature

In this section, the TAD algorithm, the TRAD algorithm, and the TAD-TRAD algorithm are compared to the existing works which compute the shortest path between two points among curved obstacles. n denotes the number of arcs. Let h denote the number of curved obstacles. Table 2 compares the TAD algorithm, the TRAD algorithm, and the TAD-TRAD algorithm with the existing polynomial time works in literature based on the time complexity, nature of the path computed (optimal or approximate), and the support for non-convex obstacles.

Table 2 indicates that the TAD algorithm, the TRAD algorithm, and the TAD-TRAD algorithm supports non-convex obstacles. Hence, the curved obstacles can be convex or non-convex.

8.2.1. Approximate shortest path algorithms

Table 2 shows that the TRAD algorithm has the least time complexity ($O(n \log n)$) followed by Hershberger et al. [14] ($O(n \log n + n \log 1/\epsilon)$) for computing an approximate path among the curved obstacles. Therefore, the TRAD algorithm can be used for the computation of the approximate path for a mobile robot among the curved obstacles.

8.2.2. Optimal shortest path algorithms

Table 2 shows that the time complexity of the TAD algorithm, the TAD-TRAD algorithm, Chen et al. [6], and Bharath et al. [30] are $O(n^3 \log n)$ or $O(h^2 n \log n), O(n(\log n)^2)$ or $O(n \log n + h \log h \log n), O(n \log n + h^2)$ or $O(n + h \log^{1+\epsilon} h + h^2)$, and $O(n^4)$ respectively. The TAD algorithm has a lower time complexity than the existing implemented algorithms for the optimal shortest path computation. The TAD-TRAD algorithm has the least

Table 2
Comparison of the TAD algorithm, the TRAD algorithm, and the TAD-TRAD algorithm with the existing polynomial time works in literature.

Algorithm	Time Complexity	Nature of the path computed	Non-Convex obstacles	Theoretical/ Implemented
Pocchiola et al. [26]	$O(n^2 \log n + n^2)$	Optimal	Not supported	Theoretical
Chen et al. [5]	$O(n \log n + n^2)$	Optimal	Not supported	Theoretical
Chen et al. [6]	$O(n \log n + h^2)$ or $O(n + h \log^{1+\epsilon} h + h^2)$	Optimal	Supported	Theoretical
Hershberger et al. [14]	$O(n \log n + n \log 1/\epsilon)$	Approximate	Supported	Theoretical
Bharath et al. [30]	$O(n^4)$	Optimal	Supported	Implemented
TAD	$O(n^3 \log n)$ or $O(h^2 n \log n)$	Optimal	Supported	Implemented
TRAD	$O(n \log n)$	Approximate	Supported	Implemented
TAD-TRAD	$O(n(\log n)^2)$ or $O(n \log n + h \log h \log n)$	Optimal	Supported	Implemented

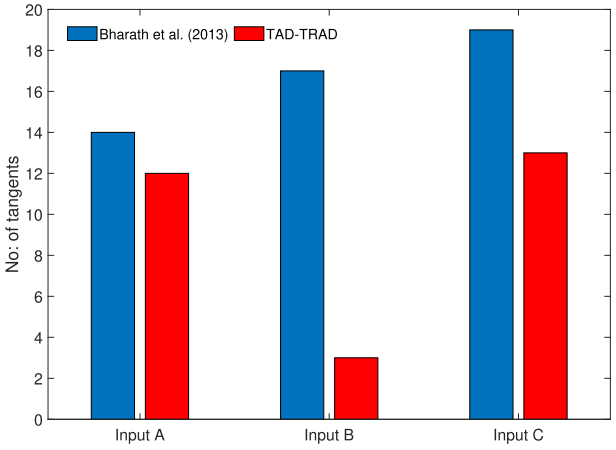


Fig. 11. Comparison of the number of tangents of the TAD-TRAD algorithm with Bharath et al. [30]. The input problems are taken from [30].

time complexity when compared to all existing algorithms for the optimal shortest path computation among curved obstacles.

Similarly, the number of tangents constructed in the TAD algorithm, the TAD-TRAD algorithm, Chen et al. [6], and Bharath et al. [30] are $O(n^2)$ or $O(h^2), O(n\log n)$ or $O(h\log h), O(h^2)$, and $O(n^2+nh+n+h)$ respectively. Among the existing algorithms, Chen et al. [6] has the lowest number of tangents. The number of tangents constructed in the TAD algorithm is equal to the number of tangents constructed in Chen et al. [6]. The number of tangents constructed in the TAD-TRAD algorithm is lesser than all existing algorithms for the optimal shortest path computation among curved obstacles.

Fig. 11 shows the comparison of the number of tangents of the TAD-TRAD algorithm with Bharath et al. [30], which is the only existing implemented algorithm. The input problems are taken from [30]. It can be seen that the number of tangents used for the TAD-TRAD algorithm is lesser than that of [30]. Note that the number of tangents used for the TAD algorithm is same as the TAD-TRAD algorithm and is not explicitly depicted in Fig. 11. The running time of the TAD-TRAD algorithm for Input A, Input B, and Input C are 0.066 s, 0.02 s, and 0.07 s respectively. The TAD-TRAD algorithm computes the shortest path using a lesser number of tangents in a small amount of time.

The TAD-TRAD algorithm generates an optimal path with the lowest time complexity and the least number of tangents, compared to existing algorithms. Thus, the TAD-TRAD algorithm is advantageous over existing algorithms for computing the shortest optimal path for a mobile robot between two points among curved obstacles.

8.2.3. Comparison of TAD-TRAD algorithm with stochastic optimization algorithms in literature

In this Section, the TAD-TRAD algorithm is compared with stochastic optimization algorithms in literature in order to show the efficiency of the TAD-TRAD algorithm. The TAD-TRAD algorithm is compared with [3,1,32,19] for the path optimization among circular obstacles. Note that [3,1,32,19] works for only circular obstacles.

[3] uses a hybridized Particle Swarm Optimization-Modified Frequency Bat (Hybrid PSO-MFB) algorithm, [1] uses Direct Artificial Bee Colony (DABC) algorithm and Minimum Angle Artificial Bee Colony (MAABC) algorithm, [32] uses genetic algorithm (GA) and Bacterial Colony (BC) algorithm, and [19] uses Standard Artificial Bee Colony (ABC) algorithm.

Fig. 12 shows the comparison of distance in meters of shortest path computed by the TAD-TRAD algorithm and the Hybrid PSO-MFB algorithm [3]. The input problem is taken from [3] and it consists of five circular obstacles. It can be seen that the distance computed by the TAD-TRAD algorithm is lesser than that of the Hybrid PSO-MFB algorithm [3].

Fig. 13 shows the comparison of distance in meters of shortest path computed by the TAD-TRAD algorithm, the Hybrid PSO-MFB algorithm [3], DABC and MAABC algorithms [1], and GA and BC algorithms [32]. The input problem is taken from [3] and it consists of four circular obstacles. It can be seen that the distance computed by the TAD-TRAD algorithm is lesser than that of the algorithms proposed in [3,1,32].

Fig. 14 shows the comparison of distance in meters of shortest path computed by the TAD-TRAD algorithm, the Hybrid PSO-MFB algorithm [3], DABC and MAABC algorithms [1], and ABC algorithm [19]. The input problem is taken from [19] and it consists of six circular obstacles. It can be seen that the distance computed by the TAD-TRAD algorithm is lesser than that of the algorithms proposed in [3,1,19].

Note that the TAD-TRAD algorithm is a polynomial time algorithm which always guarantees of finding the global optimum whereas the stochastic algorithms [3,1,32,19] have no guarantee of finding the global optimum in a finite amount of time.

9. Conclusion

This work proposes three new algorithms (the TAD algorithm, the TRAD algorithm, and the TAD-TRAD algorithm) to compute

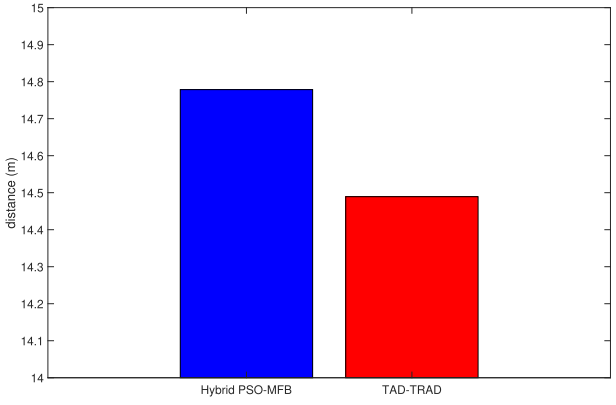


Fig. 12. Comparison of distance (m) of shortest path computed by the TAD-TRAD algorithm and the Hybrid PSO-MFB algorithm [3]. The input problem is taken from [3].

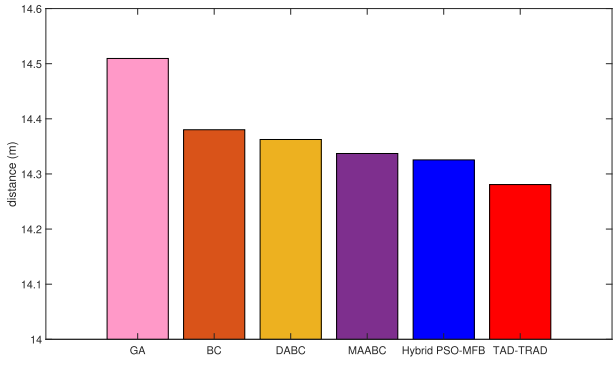


Fig. 13. Comparison of distance (m) of shortest path computed by the TAD-TRAD algorithm, the Hybrid PSO-MFB algorithm [3], DABC and MAABC algorithms [1], and GA and BC algorithms [32]. The input problem is taken from [3].

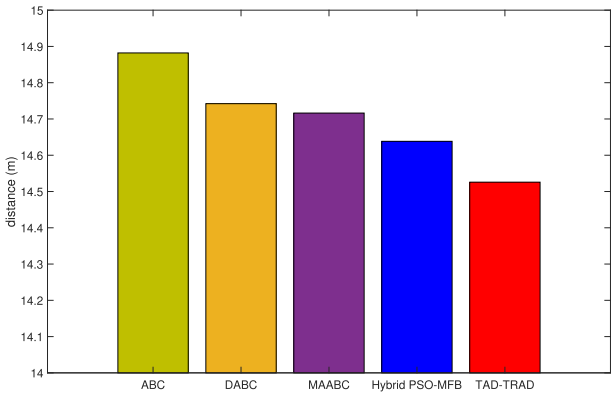


Fig. 14. Comparison of distance (m) of shortest path computed by the TAD-TRAD algorithm, the Hybrid PSO-MFB algorithm [3], DABC and MAABC algorithms [1], and ABC algorithm [19]. The input problem is taken from [19].

the shortest path for a mobile robot between a source point S and an end point E among curved obstacles. The TAD algorithm, the TRAD algorithm, and the TAD-TRAD algorithm has a time complexity of $O(n^3 \log n), O(n \log n)$, and $O(n(\log n)^2)$ respectively, where n is the total number of arcs in all the obstacles. Both the TAD algorithm and the TAD-TRAD algorithm generates an optimal path, whereas the TRAD algorithm generates an approximate path.

Seciton 8.1 compares the performance of the TAD algorithm, the TRAD algorithm, and the TAD-TRAD algorithm for the 200 randomly generated problems. The TAD-TRAD algorithm generates the shortest path in the least amount of time.

Seciton 8.2 compares the TAD algorithm, the TRAD algorithm, and the TAD-TRAD algorithm with the existing works in literature. The TAD algorithm has a smaller time complexity than all existing implemented algorithms for the optimal shortest path computation. The number of tangents constructed in the TAD algorithm is same as that of the best existing algorithm for the optimal shortest path computation. The TRAD algorithm has a smaller time complexity than all existing algorithms for approximate shortest path computation between two points among curved obstacles. The TAD-TRAD algorithm has a smaller time complexity and smaller number of tangents than all existing algorithms for optimal shortest path computation between two points among curved obstacles.

The comparison with 6 problems in the literature shows that the TAD-TRAD algorithm computes an optimal path in lesser time. Moreover, the number of tangents constructed for the shortest path computation is lesser in the TAD-TRAD algorithm than existing algorithms. Therefore, the TAD-TRAD algorithm can be used for computing the optimal path for a mobile robot between two points among curved obstacles in less time.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] N.H. Abbas, F.M. Ali, Path planning of an autonomous mobile robot using directed artificial bee colony algorithm, Int. J. Comput. Appl. 96 (2014).
[2] M. Abrahamsen, B. Walczak, Common tangents of two disjoint polygons in linear time and constant workspace, ACM Trans. Algorithms (TALG) 15 (2018) 1–21.
[3] F.H. Ajeil, I.K. Ibraheem, M.A. Sahib, A.J. Humaidi, Multi-objective path planning of an autonomous mobile robot using hybrid pso-mfb optimization algorithm, Appl. Soft Comput. 89 (2020) 106076.
[4] A. Blake, M. Brady, R. Cipolla, Z. Xie, A. Zisserman, Visual navigation around curved obstacles, in: Proceedings. 1991 IEEE International Conference on Robotics and Automation, IEEE Computer Society, 1991, pp. 2490–2491..
[5] D.Z. Chen, H. Wang, Computing shortest paths amid pseudodisks, in: Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete algorithms, SIAM, 2011, pp. 309–326..
[6] D.Z. Chen, H. Wang, Computing shortest paths among curved obstacles in the plane, ACM Trans. Algorithms (TALG) 11 (2015) 1–46.
[7] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to algorithms, MIT press, 2009.
[8] M. De Berg, M. Van Kreveld, M. Overmars, O. Schwarzkopf, Computational geometry, in: Computational geometry, Springer, 1997, pp. 1–17.
[9] D.P. Dobkin, D.L. Souvaine, Computational geometry in a curved world, Algorithmica 5 (1990) 421–457.
[10] D.P. Dobkin, D.L. Souvaine, C.J. Van Wyk, Decomposition and intersection of simple splinegons, Algorithmica 3 (1988) 473–485.
[11] N.A. Eapen, A.A. Beegom, A linear time pre-processing for optimization of shortest path and distance algorithms, in: 2017 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES), IEEE, 2017, pp. 1–5.
[12] S.K. Ghosh, D.M. Mount, An output-sensitive algorithm for computing visibility graphs, SIAM J. Comput. 20 (1991) 888–910.
[13] J. Hershberger, S. Suri, An optimal algorithm for euclidean shortest paths in the plane, SIAM J. Comput. 28 (1999) 2215–2256.
[14] J. Hershberger, S. Suri, H. Yıldız, A near-optimal algorithm for shortest paths among curved obstacles in the plane, in: Proceedings of the twenty-ninth annual symposium on Computational geometry, 2013, pp. 359–368..
[15] S. Kapoor, S. Maheshwari, Efficient algorithms for euclidean shortest path and visibility problems with polygonal obstacles, in: Proceedings of the fourth annual symposium on Computational geometry, 1988, pp. 172–182.
[16] S. Kapoor, S. Maheshwari, J.S. Mitchell, An efficient algorithm for euclidean shortest paths among polygonal obstacles in the plane, Discrete Computat. Geometry 18 (1997) 377–383.
[17] S. Kim, S. Bhattacharya, V. Kumar, Path planning for a tethered mobile robot, in: 2014 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2014, pp. 1132–1139.
[18] Y. Li, W. Wei, Y. Gao, D. Wang, Z. Fan, Pq-rrt*: an improved path planning algorithm for mobile robots, Expert Syst. Appl. (2020) 113425.
[19] J.H. Lin, L.R. Huang, et al., Chaotic bee swarm optimization algorithm for path planning of mobile robots, in: Proceedings of the 10th WSEAS international conference on evolutionary computing, World Scientific and Engineering Academy and Society (WSEAS), 2009, pp. 84–89.
[20] H. Ling, D.W. Jacobs, Shape classification using the inner-distance, IEEE Trans. Pattern Anal. Mach. Intell. 29 (2007) 286–299.
[21] Y.H. Liu, S. Arimoto, Proposal of tangent graph and extended tangent graph for path planning of mobile robots, in: Proceedings. 1991 IEEE International Conference on Robotics and Automation, IEEE Computer Society, 1991, pp. 312–313.
[22] Y.H. Liu, S. Arimoto, Path planning using a tangent graph for mobile robots among polygonal and curved obstacles: communication, Int. J. Robot. Res. 11 (1992) 376–382.
[23] H. Martınez-Alfaro, S. Gomez-Garcıa, Mobile robot path planning and tracking using simulated annealing and fuzzy logic control, Expert Syst. Appl. 15 (1998) 421–429.
[24] E.A. Melissaratos, D.L. Souvaine, Shortest paths help solve geometric optimization problems in planar regions, SIAM J. Comput. 21 (1992) 601–638.
[25] J.S. Mitchell, Shortest paths among obstacles in the plane, Int. J. Comput. Geometry Appl. 6 (1996) 309–332.
[26] M. Pocchiola, G. Vegter, Topologically sweeping visibility complexes via pseudotriangulations, Discrete Comput. Geometry 16 (1996) 419–453.
[27] Y.Q. Qin, D.B. Sun, N. Li, Y.G. Cen, Path planning for mobile robot using the particle swarm optimization with mutation operator, in: Proceedings of 2004 international conference on machine learning and cybernetics (IEEE Cat. No. 04EX826), IEEE, 2004, pp. 2473–2478..
[28] S.B. Ram, M. Ramanathan, The shortest path in a simply-connected domain having a curved boundary, Comput. Aided Des. 43 (2011) 923–933.
[29] S.B. Ram, M. Ramanathan, Comment on clarification on the short communication on computing the shortest path in a multiply-connected domain having curved boundaries, Comput. Aided Des. (2014) 64–65.
[30] M. Ramanathan et al., Shortest path in a multiply-connected domain having curved boundaries, Comput. Aided Des. 45 (2013) 723–732.
[31] R. Sarkar, D. Barman, N. Chowdhury, Domain knowledge based genetic algorithms for mobile robot path planning having single and multiple targets, J. King Saud Univ.-Comput. Inform. Sci., 2020..
[32] C.A. Sierakowski, L.D.S. Coelho, Study of two swarm intelligence techniques for path planning of mobile robots, in: 16th IFAC World Congress, Prague, 2005.
[33] B. Song, Z. Wang, L. Zou, An improved pso algorithm for smooth path planning of mobile robots using continuous high-degree bezier curve, Appl. Soft Comput. 100 (2020) 106960.
[34] J.A. Storer, J.H. Reif, Shortest paths in the plane with polygonal obstacles, J. ACM (JACM) 41 (1994) 982–1012.
[35] J. Wang, M.Q.H. Meng, Optimal path planning using generalized voronoi graph and multiple potential functions, IEEE Trans. Industr. Electron. (2020).
[36] X. Wei, A. Joneja, On computing the shortest path in a multiply-connected domain having curved boundaries, Comput. Aided Des. 48 (2014) 39–41.