




MOD-RRT*: A Sampling-Based Algorithm for Robot Path Planning in Dynamic Environment

Jie Qi , Hui Yang , and Haixin Sun , *Member, IEEE*

Abstract—This article presents an algorithm termed as multiobjective dynamic rapidly exploring random (MOD-RRT*), which is suitable for robot navigation in unknown dynamic environment. The algorithm is composed of a path generation procedure and a path replanning one. First, a modified RRT* is utilized to obtain an initial path, as well as generate a state tree structure as prior knowledge. Then, a shortcutting method is given to optimize the initial path. On this basis, another method is designed to replan the path if the current path is infeasible. The suggested approach can choose the best node among several candidates within a short time, where both path length and path smoothness are considered. Comparing with other static planning algorithms, the MOD-RRT* can generate a higher quality initial path. Simulations on the dynamic environment are conducted to clarify the efficient performance of our algorithm in avoiding unknown obstacles. Furthermore, real applicative experiment further proves the effectiveness of our approach in practical applications.

Index Terms—Autonomous mobile robot, dynamic path planning, multiobjective planning, rapidly exploring random tree (RRT).

I. INTRODUCTION

IN RECENT years, an autonomous mobile robot plays a significant role in various fields, such as emergency rescue, autonomous exploitation, storehouse management, and so forth. As a fundamental research topic in robotics, path planning has been received considerable attention. Generally speaking, the path planning problem can be defined by determining a collision-free trajectory between the robot's start position and its destination. In addition, trajectory optimization is also important to the robot because of the requirement in energy optimal and time optimal. Path smoothing schemes using the spline curves are proposed [1]–[3]. In [4], the terrain roughness is considered during the planning for less energy consumption.

Manuscript received January 8, 2020; revised April 13, 2020; accepted May 19, 2020. Date of publication June 11, 2020; date of current version April 27, 2021. This work was supported in part by the National Key R&D Program of China under Grant 2018YFC0809200 and in part by the National Natural Science Foundation of China under Grants 61671394 and 61971362 (*Corresponding author: Jie Qi.*)

Jie Qi and Hui Yang are with the College of Electronic Science and Technology, Xiamen University, Xiamen 361005, China (e-mail: qjie@xmu.edu.cn; hyang@stu.xmu.edu.cn).

Haixin Sun is with the School of Informatics, Xiamen University, Xiamen 361005, China (e-mail: hxsun@xmu.edu.cn).

Color versions of one or more of the figures in this article are available online at <https://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TIE.2020.2998740

Sampling-based algorithms, such as rapidly exploring random tree (RRT) [5] and probabilistic roadmap algorithm [6], have proven to be effectively in solving many tough planning problems, particularly with nonlinear systems. In robot path planning, these methods are often used for nonholonomic wheeled robots. However, the original RRT only focus on the speed in finding planning solutions, with less regard with the solution optimality. Karaman *et al.* [7] proved that the RRT is not asymptotically optimal and developed an asymptotically optimal RRT-based path planning scheme called optimal rapidly exploring random tree (RRT*), which makes use of a new step to rewire the neighboring vertices of the newly inserted node. The RRT* is considered as a typical extension of the RRT, and has been applied in several scenarios [8], [9]. A variety of algorithms have been proposed to improve the performance of the RRT*. To reduce the running time, the lower bound tree-RRT was proposed in [10], which is on the basis of the RRT* and rapidly exploring random graph, an approximation factor is developed to determine the state of the algorithm. Li *et al.* [11] developed a near-optimal RRT, which utilizes a neural network to predict the cost function, and designs a new reconstruction method for the random search tree. The dual tree-RRT* (DT-RRT*) in [12] separates the extension and optimization procedure using a double-tree structure. This DT-RRT* has an original RRT to explore the unknown environments, and another modified RRT* to obtain the optimal solution.

The practical environments are usually partially known. Suppose that there are several unknown obstacles lying in somewhere, if some of them occupy the pregenerated path, the robot will collide with the obstacles during following the path. Hence, it is necessary that robot replans the path in real time based on the environmental information it observes. Medina-Santiago *et al.* [13], designed an improved neural control scheme for robot to avoid hindrance in real time. Franzè *et al.* [14]. Advanced a receding horizon control algorithm for robots modeled by the linear time-invariant system in [15], a control algorithm based on the collision cone approach is proposed to avoid stationary obstacle. Incremental search algorithms are preferable over the traditional methods in dynamic environments [16]–[18], because incremental algorithm preplans several candidate trajectories and the robot can select one of them if the current path is infeasible. The tree structure of the RRT makes it useful for incremental planning and there are already some researches using incremental algorithms based on the RRT in path planning [19], [20].

Most existing algorithms only attempt to optimize the path length. However, path smoothness is also an important property of a path [21]. Optimizing path smoothness can efficiently reduce robot's energy consumption [22]. Therefore, some multiobjective path planning schemes have been presented. A multiobjective path planning algorithm based on reinforcement learning was proposed in [23]. Oral *et al.* [24] developed a multiobjective D* Lite (MOD* Lite), which is based on the well-known D* Lite algorithm, and optimizes multiple objectives at the same time. The algorithm in [25] combines multiobjective planning and particle swarm optimization, both the risk degree and the distance of path are considered during its planning. Hence, it is necessary to develop a replanning method that can select the best node in several candidates in a short time.

In this article, the dynamic path planning problem is divided into initial path generation and path replanning during navigation. A multiobjective RRT* (MOD-RRT*) scheme is presented, which consists of two separate procedure to deal with each step. First, a modified RRT* algorithm is proposed to generate an initial path and construct a state tree structure on the basis of the known map. And a path replanning scheme is proposed to avoid the unknown obstacles. Overall, the main contributions of this article can be summarized as follows.

- 1) A backward expansion strategy is designed for the RRT*, hence, the cost value from any tree nodes to goal position can be stored as prior information.
- 2) A path optimization method is put forward to reduce the effect of random sampling on the path length and path smoothness.
- 3) A reconnection scheme is designed to optimize the tree structure under the circumstances of unknown obstacles.
- 4) A multiobjective path replanning method based on the Pareto theory is presented to select the best alternative node if obstacles block the preplanned path.

The remainder of this article is organized as follows. Section II presents the problem statements of dynamic path planning. In Section III, the main subalgorithms of the MOD-RRT* are presented in detail. Several simulations and one real experiment are presented in Section IV to show the effectiveness of the MOD-RRT*. Finally, Section V concludes this article.

II. PROBLEM STATEMENTS

In this section, the definition of the dynamic path planning problem is presented. Different from the static path planning, the map in the dynamic path planning is partially known. That is, the initial map only contains the location of some obstacles, other obstacles are not marked. The robot has a detector (e.g., lidar) within a certain detection range, to detect the unknown obstacles, where a localization device is to accurately locate itself. The working environment is depicted in Fig. 1.

To formulate the dynamic path planning problem clearly, we abstract the working environment into a configuration space. χ_{obs} denotes the obstacle region, and the free space is represented as χ_{free} . x_{init} and x_{goal} denotes the start position and the goal position. Then, the dynamic path planning problem can be expressed as: first finding an initial path $\sigma : [0, 1] \rightarrow \chi_{\text{free}}$ on the basis of

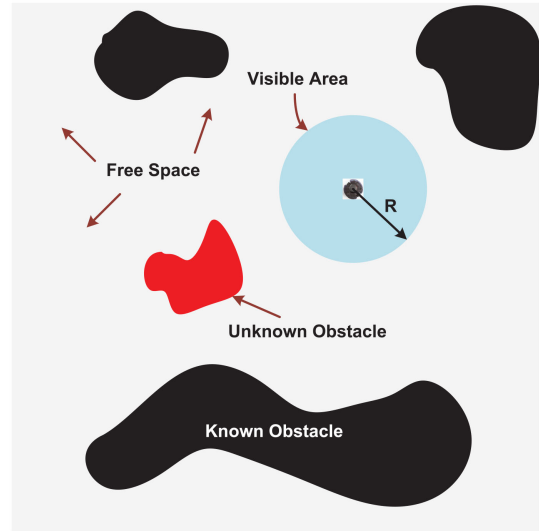


Fig. 1. Working environment of an autonomous mobile robot.

the known map, where $\sigma(0) = x_{\text{init}}$ and $\sigma(1) = x_{\text{goal}}$. During following the path, updating the map by detecting obstacles, replanning the path to ensure $\sigma(t) \in \chi_{\text{free}}$ with any $t \in [0, 1]$.

Probabilistic completeness and asymptotic near-optimality property are two common indicators in path planning, which is defined as follows.

Definition 1 (Probabilistic completeness): Given a path planning problem, probabilistic completeness means an algorithm can eventually find a feasible path if it exists.

Definition 2 (Asymptotic near-optimality): Given a path planning problem, the cost function of a trajectory is $c(\sigma)$. The asymptotic near-optimality means if the number of samples tend to infinity, $c(\sigma)$ converges to the optimal value.

Since the map is partially known, the requirements of the dynamic path planning algorithm is that: keeping asymptotic near-optimality property and probabilistic completeness in generating the initial path, and minimizing the impact on the path quality if replanning.

III. MULTIOBJECTIVE DYNAMIC RRT* (MOD-RRT*)

In this section, the (MOD-RRT*) is proposed for dynamic path planning. In the MOD-RRT*, first a modified RRT* algorithm is presented to generate an initial path and a state tree on the basis of the known map. During following the initial path, the global map and the state tree structure are modified, and a Pareto dominance path replanning method is presented to avoid unknown obstacles. These two algorithms are described as follows in detail.

A. Initial Path Generation

Before navigation, an initial path is generated on the basis of the known map. $T.V$ and $T.E$ are defined as the set of vertices of the tree and edges, respectively. Different from the traditional RRT* methods, the state tree expands with the goal position as the root node. In this way, the cost value from any tree node to goal position can be calculated exactly, which helps select better

Algorithm 1: Path-generation Function.

```

1:  $T.V \leftarrow x_{\text{goal}};$ 
2:  $T.E \leftarrow \text{null};$ 
3: while Build_Tree do
4:    $x_{\text{rand}} \leftarrow \text{Rand\_Sample};$ 
5:    $x_{\text{nearest}} \leftarrow \text{Nearest}(x_{\text{rand}}, T.V);$ 
6:    $x_{\text{new}} \leftarrow \text{Steer}(x_{\text{nearest}}, x_{\text{rand}});$ 
7:   if ( $\neg \text{Collision\_Free}(x_{\text{nearest}}, x_{\text{new}})$ ) then
8:     Continue;
9:   end if
10:   $x_{\text{parent}} \leftarrow x_{\text{nearest}};$ 
11:   $\text{Cost}(x_{\text{new}}) \leftarrow \text{Cost}(x_{\text{nearest}}) + d(x_{\text{nearest}}, x_{\text{new}});$ 
12:   $X_{\text{near}} \leftarrow \text{NeighborNodes}(x_{\text{new}}, T.V, d);$ 
13:  for  $x_{\text{near}} \in X_{\text{near}}$  do
14:     $x_{\text{parent}} \leftarrow \text{Rewire}(x_{\text{near}}, x_{\text{new}});$ 
15:  end for
16:   $T.V \leftarrow T.V \cup (x_{\text{new}});$ 
17:   $T.E \leftarrow T.E \cup (x_{\text{parent}}, x_{\text{new}});$ 
18:  for  $x_{\text{near}} \in X_{\text{near}}$  do
19:     $x_{\text{parenttemp}} \leftarrow \text{Rewire}(x_{\text{new}}, x_{\text{near}});$ 
20:    if  $x_{\text{parenttemp}} = x_{\text{new}}$  then
21:       $T.E \leftarrow T.E - (x_{\text{parent}} \text{ of } x_{\text{near}}, x_{\text{near}});$ 
22:       $T.E \leftarrow T.E \cup (x_{\text{new}}, x_{\text{near}});$ 
23:    end if
24:  end for
25: end while

```

nodes when replanning the path. The pseudocode of the initial path generation is described in Algorithm 1.

Terminologies we used throughout this article are defined as follows.

Rand_Sample: A procedure returns a new state x_{rand} , which is generated at random.

Nearest($x, T.V$), NeighborNodes($x, T.V, d$): Procedure returns the nearest node of x in $T.V$, and nodes in $T.V$ whose Euclidean distance to x is less than d .

Steer(x_1, x_2): A procedure returns a new node x steering from x_1 to x_2 at a certain distance.

Collision_Free(x_1, x_2): A procedure tests if the straight-line segment from x_1 to x_2 is feasible.

Cost(x): A procedure returns the total path length from node x to x_{goal} .

$d(x_1, x_2)$: A procedure returns the Euclidean distance from x_1 to x_2 .

Pareto(X): A procedure returns the best node among the set of nodes X using the Pareto dominance theory.

After x_{new} is added to the tree, its current parent node is x_{nearest} . To maintain the asymptotic optimality property, a rewire process is proceeded. The set X_{near} is the neighbor nodes of x_{new} within the rewire range d . For each node $x_{\text{near}} \in X_{\text{near}}$, the node that minimizes $\text{Cost}(x_{\text{new}})$ will be chosen as the new parent node of x_{new} . And the edge from x_{parent} to x_{new} will be added in $T.E$. Afterwards, the x_{new} will be considered whether it can be the parent node of every $x_{\text{near}} \in X_{\text{near}}$ or not. The pseudocode of $\text{Rewire}(x_1, x_2)$ is described in Algorithm 2.

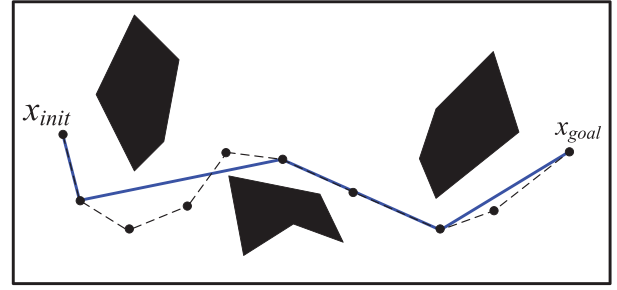


Fig. 2. Original path and the optimized path.

Algorithm 2: Rewire(x_1, x_2).

```

1: if ( $\text{Collision\_Free}(x_1, x_2)$ ) then
2:   if  $\text{Length}(x_1, x_2) + \text{Cost}(x_1) < \text{Cost}(x_2)$  then
3:      $\text{Cost}(x_2) = \text{Length}(x_1, x_2) + \text{Cost}(x_1);$ 
4:      $x_{\text{parent}} \leftarrow x_1;$ 
5:   end if
6: end if

```

When the state tree expands to x_{init} , a path consists of several tree nodes and the edges between them can be generated. Since this path usually has several redundant turning points, an ant colony optimization algorithm is presented to further optimize the path, in terms of path length and path smoothness. Assume that a path ($x_{\text{init}}, x_1, x_2, \dots, x_n, x_{\text{goal}}$), initially all ants are located in x_{init} , and each ant randomly selects its next path point as

$$P_{ij}^k = \frac{\tau_{ij}^\alpha \eta_j^\beta}{\sum_{l \in \text{allowed}(k)} \tau_{il}^\alpha \eta_l^\beta}, j \in \text{allowed}(k) \quad (1)$$

where P_{ij}^k is the probability for the ant k moving from x_i to x_j , τ_{ij} is the pheromone concentration between x_i and x_j , η_j is the heuristic information of x_j . $\text{allowed}(k)$ is the set of reachable path points. For instance, if ant k is in x_i , its reachable points x are the subsequent path points that satisfies $\text{Collision_Free}(x_i, x)$. α and β are weight values. η_j and τ_{ij} are the two parameters that determine the probability to select a path node. η_j is calculated as the inverse of the Euclidean distance from x_j to x_{goal} . τ_{ij} is initialized to 0, after all ants reach x_{goal} , τ_{ij} is updated as follows:

$$\tau_{ij}(t+1) = (1 - \rho)\tau_{ij}(t) + \rho \cdot \sum \tau_{ij}^k(t) \quad (2)$$

and

$$\tau_{ij}^k(t) = \begin{cases} \frac{1}{L + \sum \theta}, & \text{if ant } k \text{ passes } x_i \text{ and } x_j \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

where ρ is the pheromone evaporation ratio, $\tau_{ij}^k(t)$ is the pheromone deposited by the ant k between x_i and x_j . Two criteria are considered, L is the path length and $\sum \theta$ is the sum of steering angles on the path. These two criteria will be normalized in pheromone updating. When the iteration ends, the best solution will be chosen as the optimized path. Fig. 2 depicts the original path and the optimized path, obviously that the path length and steering angle has been reduced after optimization.

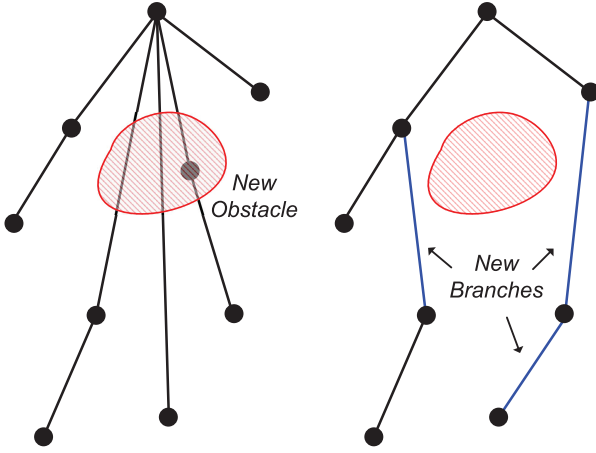


Fig. 3. Illustration of reconnection step under the presence of new obstacle.

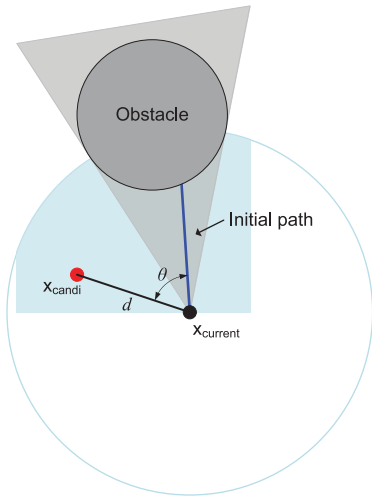


Fig. 4. Illustration of candidate nodes in path replanning.

B. Path Replanning

During navigation, the robot updates the map in real time on the basis of the environmental information it detects. Meanwhile, the state tree will optimize its structure, to make sure that its vertices and edges do not collide with new detected obstacles. The reconnection step is illustrated in Fig. 3. The colliding nodes and vertices will be deleted. The deletion may result in several orphan nodes, and these orphan nodes will select new node as their parent node. For an orphan node x_{orp} , the reconnection range is r , and the candidates of its new parent node are $\text{NeighborNodes}(x_{orp}, T.V, r)$. The node that minimizes $\text{Cost}(x_{orp})$ will be its new parent node.

When robot finds new obstacles blocking its preplanned path, it will search another node on the state tree as an alternative node to avoid obstacles. As illustrated in Fig. 4, suppose that the current location of the robot is x_{cur} , the selection range of alternative nodes is marked blue in the figure, which is the nodes within the robot detection range except the direction of obstacle.

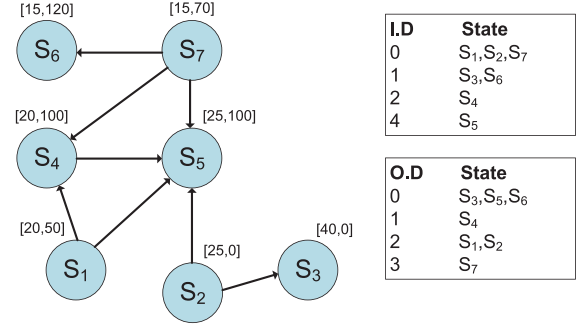


Fig. 5. Illustration of the state expansion graph.

Two indicators are considered to evaluate a candidate node x_{candi} , such as $\text{length}(x_{candi})$ and $\text{angle}(x_{candi})$. $\text{length}(x_{candi})$ is the total path length from x_{cur} to x_{goal} through x_{candi} , which is calculated as

$$\text{length}(x_{candi}) = d(x_{cur}, x_{candi}) + \text{cost}(x_{candi}) \quad (4)$$

where $d(x_{cur}, x_{candi})$ is the Euclidean distance from x_{cur} to x_{candi} . $\text{angle}(x_{candi})$ is the turning angle if the robot steering to x_{candi} , which is θ in Fig. 4.

There may be dozens to hundreds candidate nodes among the selection range. In order to select the best node, the Pareto dominance [26] is used to make pairwise comparisons between nodes, which is defined as follows.

Definition 3 (Pareto dominance): For a vector $v = [v_1, v_2, \dots, v_n]$, it dominates another vector $u = [u_1, u_2, \dots, u_n]$ if $v_i \leq u_i$ for all i and $v_j < u_j$ for at least one j where $1 \leq i, j \leq n$.

Assume that $S_1 = [20, 50]$, $S_2 = [25, 0]$, and $S_3 = [25, 100]$ are objective vectors for three states. S_1 and S_2 are nondominated with each other, whereas they are both dominated with S_3 .

With Pareto dominance, the candidate nodes can be prioritized. In path replanning, the elements of objective vector are $\text{length}(x_{candi})$ and $\text{angle}(x_{candi})$. As illustrated in Fig. 5, a directed acyclic state expansion graph is used to topologically order candidate nodes. These nodes are each represented by a state and its value. For every two states, if one state dominates another, an edge is introduced from this state to the states it dominates. There are no edge between the states that are nondominated with each other. As a result, for a state S , the in-degree of S represents the number of states that dominates S , and the outgoing degrees of S correspond to the number of states that are dominated by S . Node with in-degree 0 are nondominated states (e.g., Pareto optimal states). The number of Pareto optimal states is commonly more than one, and the state with the highest out-degree will be selected as the best state. If the number is still more than one, one of them will be selected randomly. In Fig. 5, the in-degrees and out-degrees of all states are given as a list. S_7 is the best state with its in-degree is 0 and out-degree is 3. Fig. 6 illustrates the graph after adding a new state S_8 . S_8 dominates S_1 , S_4 , and S_5 and no state dominates S_8 , hence, its in-degree is 0 and out-degree is 3. Besides, the

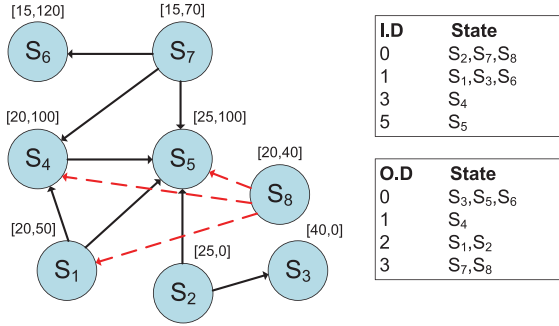


Fig. 6. Illustration of the state expansion graph after adding S_8 .

Algorithm 3: Path Replanning Function.

```

1: while NotReachGoal do
2:   if FindObstacle then
3:      $x_{cur} \leftarrow \text{CurrentPosition};$ 
4:      $X_{candi} \leftarrow \emptyset;$ 
5:      $X_{near} \leftarrow \text{NeighborNodes}(x_{cur}, T.V, d);$ 
6:     for  $x_{near} \in X_{near}$  do
7:       if Collision-Free( $x_{cur}, x_{near}$ ) then
8:          $X_{candi} \leftarrow X_{candi} \cup x_{near};$ 
9:       end if
10:    end for
11:     $x_{next} \leftarrow \text{Pareto}(X_{candi});$ 
12:  end if
13:   $x_{cur} \leftarrow x_{next};$ 
14: end while

```

in-degrees of S_1 , S_4 , and S_5 all increase 1. Now, S_7 and S_8 have the same in-degree and out-degree. Hence, the best state will be generated randomly between them.

IV. SIMULATION AND EXPERIMENT

In this section, simulation and experiment are conducted to illustrate the feasibility and merit of the proposed algorithm. In the first simulation, the initial path generation method is compared with other static path planning algorithms, another simulation is conducted to validate the effectiveness of the proposed algorithm in dynamic environments. Simulations are conducted in MATLAB R2019b, a computer with Intel Core CPU i7-8700, 3.20 GHz, and 8-GB memory is used. Furthermore, the MOD-RRT* is applied on a robot for navigation to verify its effectiveness for challenging practical applications.

A. Comparison in Initial Path Generation

The first simulation is conducted to validate the effectiveness of the proposed algorithm in static path planning (e.g., initial path generation). Direct D-RRT* [27] is selected for comparison, which is an improvement of the RRT* algorithm by inserting a new heuristic function. Two maps are selected, called map Dense and map Gap, and both map size is 700×700 . The mobile robot is considered as a point. Two indicators are used to evaluate the performance: path length and path smoothness.

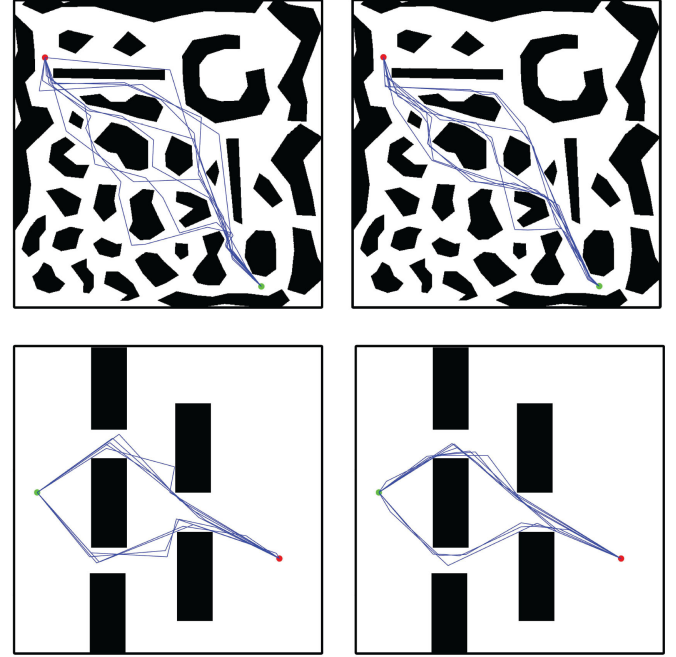


Fig. 7. Experimental results in two maps. Left: RRT*N. Right: MOD-RRT*.

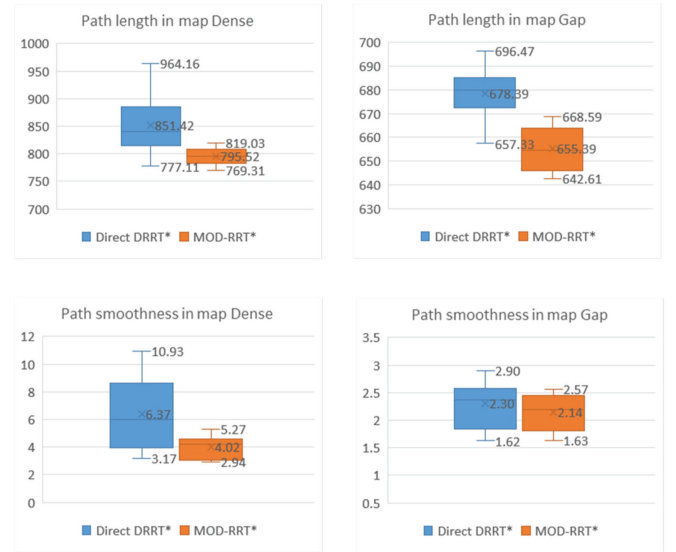


Fig. 8. Path length and path smoothness in two maps.

The smoothness is the sum of the turning angles in the path. We simulate ten times in each map.

Fig. 7 illustrates the path generated by each algorithm, and Fig. 8 shows the maximum, minimum, and average of the two parameters in ten simulations. It is obvious that the path generated by the MOD-RRT* is better in terms of length and smoothness than the Direct D-RRT*. This superiority is more apparent in complex maps. Moreover, the results of the Direct D-RRT* is much more volatile than that of the MOD-RRT*, which verifies that the MOD-RRT* has better stability. In

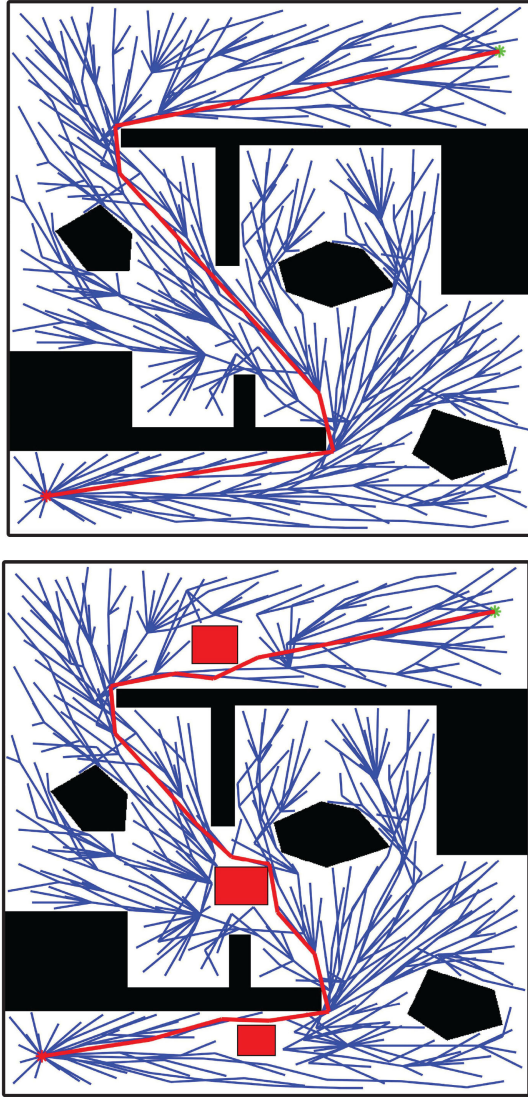


Fig. 9. Illustration of path replanning in dynamic environment. Up: Initial path. Down: Real path.

general, these results show that the MOD-RRT* is more successful in finding a high-quality feasible path.

B. Simulation Results in Dynamic Environment

The environment for the dynamic path planning simulation is a complex 2-D complex map. The map size is 700×700 . There are five known obstacles with black mark and three unknown obstacles with red mark in the map. Coordinates of start position and goal position are (650, 650) and (50, 50), respectively. The detect range of the robot is set by 70. Fig. 9 depicts the initial path and state tree structure on the basis of the known map, and the real path and reconnected tree structure during navigation. Obviously that the initial path will collide with these unknown obstacles, and the path replanning scheme successfully help the robot avoid them. During its navigation, the robot replans its path three times, that is the three bends on the real path relative to the initial path. Table I summarizes the results in three

TABLE I
SUMMARY OF RESULTS IN PATH REPLANNING

Step	Candidates	Pareto-optimal	Best node	Time(ms)
1	27	2	(276,547)	204
2	42	3	(367,312)	273
3	16	2	(294,102)	182



Fig. 10. Illustration of the Rikibot.

times path replanning, including number of candidate nodes, number of Pareto-optimal nodes, coordinates of the best node, and replanning time.

From these experimental results, the observations are as follows.

- 1) The result shows that the MOD-RRT* can generate a feasible path to help the robot effectively bypass the known and unknown obstacles.
- 2) The tree reconnecting scheme successfully optimized the tree structure to guarantee the tree nodes and edges are in the free space.
- 3) The number of candidate nodes is disparate in different locations, ranges from a dozen to dozens, and this quantity affects the execution time. But the execution time in each replanning is always less than 300 ms regardless of the quantity, which reflects the efficiency of the path replanning scheme.
- 4) The avoidance of unknown obstacles minimized the effect on the path quality, which means the path planning scheme can successfully select the best alternative node in terms of length and smoothness.

C. Real Experiment

The MOD-RRT* was further validated by real experiments. Fig. 10 depicts the experimental Rikibot, which has a lidar called Rplidar A2 to detect obstacles. The robot is driven by a Raspberry Pi, which is responsible for mapping, localization, and navigation, and a STM32MCU for robot chassis. The linear velocity and angular velocity are constrained as $v \in (0, 0.5 \text{ m/s}]$ and $\omega \in [-0.3, 0.3 \text{ rad/s}]$, respectively. The MOD-RRT* algorithm is encapsulated as a path planner in ROS navigation package. As illustrated in Fig. 11, the robot generates a laser point cloud map on the basis of the initial environment at first,

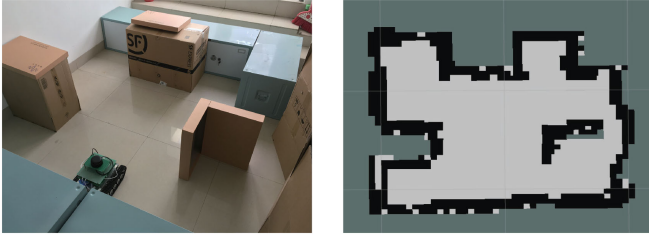


Fig. 11. Initial environment and its point cloud map.

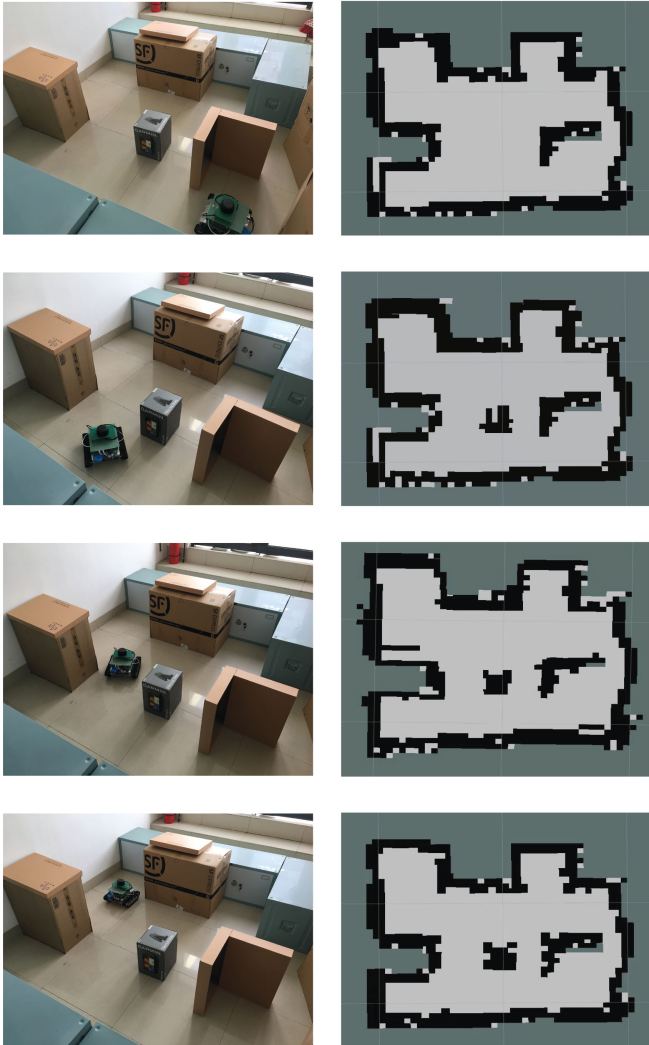


Fig. 12. Snapshots of path replanning and changes of laser data.

and stores this map as its known map. Then, an unknown obstacle is added to the environment, the robot is placed at the start position and receives instructions to move to the goal position. Fig. 12 depicts the snapshots of the path replanning and changes of the laser data, it can be seen that the robot successfully replans its path and avoids unknown obstacles. The time that robot complete its navigation is 26 s, and the linear and angular velocity during navigation are depicted in Fig. 13. Fig. 14 illustrates the trajectory of the robot. The experimental results show that the MOD-RRT* is effective in practical applications.

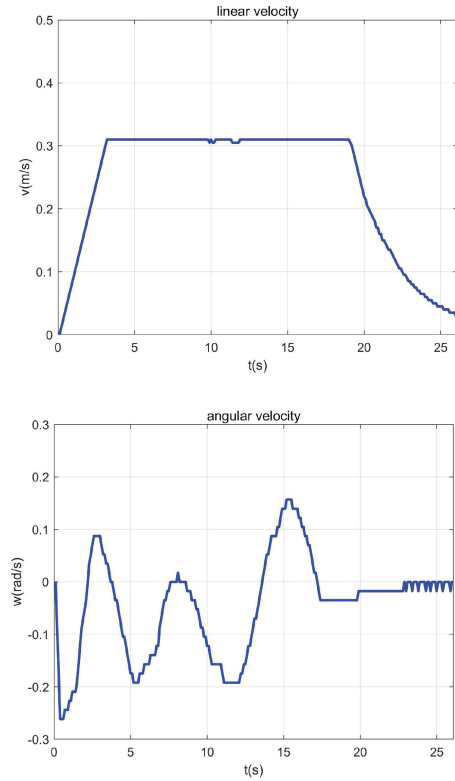


Fig. 13. Linear and angular velocity of the robot.

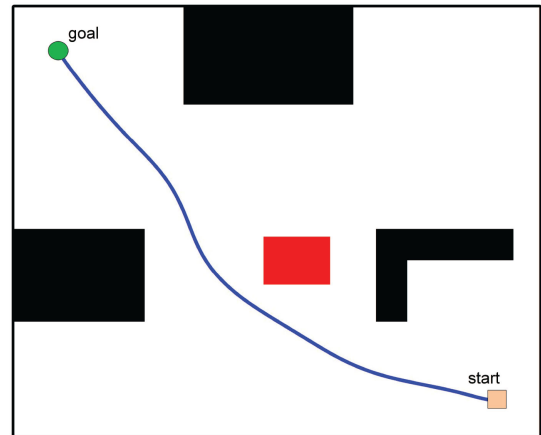


Fig. 14. Illustration of the trajectory of the robot.

V. CONCLUSION

This article was concerned of robot path planning in dynamic environment. A MOD-RRT* scheme was proposed, which had an initial path planner and a path replanner. The initial path planner was used to generate a feasible path on the basis of known map, simultaneously a state tree was also generated and stored as prior information. When unknown obstacles block the current path, the path replanner could select an alternative tree node to avoid collision. Comparing with other algorithms, it showed that the proposed algorithm could present a better solution than others in terms of path length and smoothness, as well as better stability. Simulations in dynamic environment

showed that the MOD-RRT* can select a safe node within 0.5 s when replanning the path. Real experiments also verified its effectiveness in practical applications. Our future research topic would be avoiding moving obstacles and developing a complete autonomous navigation robot system.

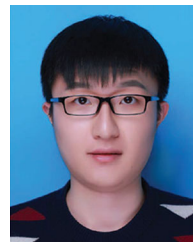
REFERENCES

- [1] C. Tsai, H. Huang, and C. Chan, "Parallel elite genetic algorithm and its application to global path planning for autonomous robot navigation," *IEEE Trans. Ind. Electron.*, vol. 58, no. 10, pp. 4813–4821, Oct. 2011.
- [2] Y. Li, T. Huang, and D. G. Chetwynd, "An approach for smooth trajectory planning of high-speed pick-and-place parallel robots using quintic b-splines," *Mech. Mach. Theory*, vol. 126, pp. 479–490, 2018.
- [3] S. MahmoudZadeh, A. Yazdani, K. Sammut, and D. Powers, "Online path planning for AUV rendezvous in dynamic cluttered undersea environment using evolutionary algorithms," *Appl. Soft Comput.*, vol. 70, pp. 929–945, 2018.
- [4] B. Wang, S. Li, J. Guo, and Q. Chen, "Car-like mobile robot path planning in rough terrain using multi-objective particle swarm optimization algorithm," *Neurocomputing*, vol. 282, pp. 42–51, 2018.
- [5] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," 1998.
- [6] L. E. Kavraki, P. Svestka, J. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.
- [7] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.
- [8] F. S. Hover et al., "Advanced perception, navigation and planning for autonomous in-water ship hull inspection," *Int. J. Robot. Res.*, vol. 31, no. 12, pp. 1445–1464, 2012.
- [9] P. Pharpata, B. Hérisse, R. Pepy, and Y. Bestaoui, "Shortest path for aerial vehicles in heterogeneous environment using RRT*," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2015, pp. 6388–6393.
- [10] O. Salzman and D. Halperin, "Asymptotically near-optimal RRT for fast, high-quality motion planning," *IEEE Trans. Robot.*, vol. 32, no. 3, pp. 473–483, Jun. 2016.
- [11] Y. Li, R. Cui, Z. Li, and D. Xu, "Neural network approximation based near-optimal motion planning with kinodynamic constraints using RRT," *IEEE Trans. Ind. Electron.*, vol. 65, no. 11, pp. 8718–8729, Nov. 2018.
- [12] L. Chen, Y. Shan, W. Tian, B. Li, and D. Cao, "A fast and efficient double-tree rrt*-like sampling-based planner applying on mobile robotic systems," *IEEE/ASME Trans. Mechatronics*, vol. 23, no. 6, pp. 2568–2578, Dec. 2018.
- [13] A. Medina-Santiago, J. Camas-Anzueto, J. Vazquez-Feijoo, H. R. Hernández-De León, and R. Mota-Grajales, "Neural control system in obstacle avoidance in mobile robots using ultrasonic sensors," *J. Appl. Res. Technol.*, vol. 12, no. 1, pp. 104–110, 2014.
- [14] G. F. e and W. Lucia, "The obstacle avoidance motion planning problem for autonomous vehicles: A low-demanding receding horizon control scheme," *Syst. Control Lett.*, vol. 77, pp. 1–10, 2015.
- [15] G. R. Mallik and A. Sinha, "A novel obstacle avoidance control algorithm in a dynamic environment," in *Proc. IEEE Symp. Comput. Intell. Secur. Def. Appl.*, Apr. 2013, pp. 57–63.
- [16] Y. Lu, X. Huo, O. Arslan, and P. Tsiotras, "Incremental multi-scale search algorithm for dynamic path planning with low worst-case complexity," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 41, no. 6, pp. 1556–1570, Dec. 2011.
- [17] S. Karaman and E. Frazzoli, "Sampling-based optimal motion planning for non-holonomic dynamical systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 5041–5047.
- [18] Y.-X. Shan, B.-J. Li, X. Guo, J. Zhou, and L. Zheng, "A considering lane information and obstacle-avoidance motion planning approach," in *Proc. 17th Int. IEEE Conf. Intell. Transp. Syst.*, Oct. 2014, pp. 16–21.
- [19] C. Moon and W. Chung, "Kinodynamic planner dual-tree RRT (DT-RRT) for two-wheeled mobile robots using the rapidly exploring random tree," *IEEE Trans. Ind. Electron.*, vol. 62, no. 2, pp. 1080–1090, Feb. 2015.
- [20] H.-I. Lin and C.-S. Yang, "2D-span resampling of bi-RRT in dynamic path planning," *Int. J. Autom. Smart Technol.*, vol. 5, no. 1, pp. 39–48, 2015.
- [21] F. Kamil and K. N., "A review on motion planning and obstacle avoidance approaches in dynamic environments," *Adv. Robot. Autom.*, vol. 4, pp. 134–142, Jan. 2015.
- [22] Y. Wang, Y. Zhao, S. A. Bortoff, and K. Ueda, "A real-time energy-optimal trajectory generation method for a servomotor system," *IEEE Trans. Ind. Electron.*, vol. 62, no. 2, pp. 1175–1188, Feb. 2015.
- [23] B. Tozer, T. Mazzuchi, and S. Sarkani, "Many-objective stochastic path finding using reinforcement learning," *Expert Syst. Appl.*, vol. 72, pp. 371–382, 2017.
- [24] T. Oral and F. Polat, "MOD* Lite: An incremental path planning algorithm taking care of multiple objectives," *IEEE Trans. Cybern.*, vol. 46, no. 1, pp. 245–257, Jan. 2016.
- [25] Y. Zhang, D. W. Gong, and J. H. Zhang, "Robot path planning in uncertain environment using multi-objective particle swarm optimization," *Neurocomputing*, vol. 103, pp. 172–185, 2013.
- [26] R. Sanders, "The Pareto principle: Its use and abuse," *J. Services Marketing*, vol. 1, no. 2, pp. 37–40, 1987.
- [27] F. O. Coelho, J. P. Carvalho, M. F. Pinto, and A. L. Marcato, "Direct-DRRT*: A RRT improvement proposal," in *Proc. 13th APCA Int. Conf. Autom. Control Soft Comput.*, Jun. 2018, pp. 154–158.



Jie Qi received the B.S. degree in automatic control and the M.S. degree in electronic engineering from Northwestern Polytechnical University, Xi'an, China, in 1995 and 1999, respectively, and the Ph.D. degree in instrument science and technology from the School of Aeronautics and Astronautics, Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2011.

She is a Lecturer of Electronic Engineering with the School of Information Science and Engineering, Xiamen University, Xiamen, China. Her research interests include optical sensing and optical communication.



Hui Yang received the B.S. degree in electronic and information engineering from Soochow University, Suzhou, China, in 2017. He is currently working toward the M.S. degree in electronic and communication engineering with Xiamen University, Xiamen, China.

His main research interests include robot location and navigation.



Haixin Sun (Member, IEEE) received the B.S. and M.S. degrees in electronic engineering from the Shandong University of Science and Technology, Shandong, China, in 1999 and 2003, respectively, and the Ph.D. degree in communication engineering from the Institute of Acoustic, Chinese Academy of Science, Shanghai, in 2006.

He visited the Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT, USA, from March 2012 to April 2013. He is currently a Professor and Doctoral Tutor with the School of Informatics, Xiamen University, Xiamen, China.

Dr. Sun was the recipient of Huawei Fellowship of Xiamen University in 2010, the Faculty of Engineering Excellence Award of Xiamen University, and the Third Prize of Chinese Army Scientific and Technological in 2017. He is a member of the Institute of Electronics, Information and Communication Engineers.