Hindawi
Complexity
Volume 2021, Article ID 6633878, 10 pages
https://doi.org/10.1155/2021/6633878

*Research Article*

# NPQ-RRT*: An Improved RRT* Approach to Hybrid Path Planning

**Zihan Yu and Linying Xiang**

*School of Control Engineering, Northeastern University at Qinhuangdao, Qinhuangdao 066004, China*

Correspondence should be addressed to Linying Xiang; xianglinying@neuq.edu.cn

Received 6 December 2020; Accepted 3 February 2021; Published 17 February 2021

Academic Editor: Jianxiang Xi



In recent years, the path planning of robot has been a hot research direction, and multirobot formation has practical application prospect in our life. This article proposes a hybrid path planning algorithm applied to robot formation. The improved Rapidly Exploring Random Trees algorithm PQ-RRT* with new distance evaluation function is used as a global planning algorithm to generate the initial global path. The determined parent nodes and child nodes are used as the starting points and target points of the local planning algorithm, respectively. The dynamic window approach is used as the local planning algorithm to avoid dynamic obstacles. At the same time, the algorithm restricts the movement of robots inside the formation to avoid internal collisions. The local optimal path is selected by the evaluation function containing the possibility of formation collision. Therefore, multiple mobile robots can quickly and safely reach the global target point in a complex environment with dynamic and static obstacles through the hybrid path planning algorithm. Numerical simulations are given to verify the effectiveness and superiority of the proposed hybrid path planning algorithm.

## 1. Introduction

With the continuous deepening of network applications, especially the rapid development of the Internet of Things, big data, cloud computing, and edge computing, the integration of information and physical systems has become increasingly close. Also, the connection between the network and human society has become much more closer. Cyber-physical-social system (CPSS) includes system engineering such as embedded environment perception, dynamic analysis of human organization behavior, network communication, and network control. Such CPSSs have functions of computing, communication, precise control, remote collaboration, and autonomy. Technologies such as artificial intelligence, multiagent, and machine learning have been more widely used in the CPSS area [1–4]. As a typical representative of agents, multiple autonomous robots, including unmanned aerial vehicles (UAVs), automatic ground vehicles (AGVs), and unmanned underwater vehicles (UUVs), play an important role in military and civil fields [5–13]. To ensure robots carry out related work in our life efficiently and safely, path planning, which means finding a feasible path without collisions from the starting state to the target state, has been a hot research point in the field of mobile robot applications [14–18]. At present, path planning algorithms can be divided into global path planning algorithms and local path planning algorithms, which mainly include geometric algorithms, artificial potential field algorithms, grid-based search algorithms, and sampling-based algorithms. Among them, the sampling-based algorithm has received extensive attention because of its superior performance in high-dimensional state space. Furthermore, the probability of finding a feasible path in space approaches 1 as the sampling number approaches infinity.

Rapidly Exploring Random Tree (RRT) [19], as a representative of sampling-based path planning, has attracted wide attention of the research community. A large number of improved algorithms for RRT have emerged in the past decade. RRT-connect [20] is a dual-tree RRT algorithm. Two random trees are generated from the start point and the end

point, respectively. However, neither RRT nor RRT-connect considers the path cost; therefore, the optimality of the algorithm cannot be guaranteed. Based on the previous algorithms, the RRT* algorithm [21] was proposed, in which the cost of the path is covered. Also, the steps of selection and rewiring are added. This algorithm obtains progressive optimality, which has become a breakthrough in the development of the Rapidly Exploring Random Trees algorithm. However, the convergence speed of the algorithm has become a new problem. One of the fundamental reasons for the slow convergence speed of RRT* is its global exploration, which does not have a specific direction. To solve this problem, P-RRT* was proposed in [22]. This algorithm incorporates APF into RRT*, and the addition of APF [23] provides a direction for exploration, making P-RRT* converge faster than RRT*. In addition, another algorithm named Quick-RRT* [24] was proposed which uses the triangle inequality to optimize the process of selecting the parent node and connection. Compared with RRT*, it has a faster convergence speed. PQ-RRT* [25] combines P-RRT* and Quick-RRT*, which makes the algorithm generate a better initial solution and can quickly converge to obtain a relatively optimal solution. However, the dynamic obstacles are not considered in PQ-RRT*. Therefore, it can be only used for static path planning and still has some limitations.

In the local path planning algorithm part, the dynamic window approach (DWA) [26] and other algorithms plan the path of the mobile robot through the surrounding information collected by the sensor. However, these algorithms usually do not consider the global map information. Tang et al. proposed a high-speed USV local response obstacle avoidance based on the DWA method [27]. However, it fails to consider the global map information. It is difficult to find the optimal path in the global range using only this kind of algorithm. Based on this kind of situation, various hybrid planning algorithms have been proposed [28–30].

Motivated by the above discussions, we improve the traditional PQ-RRT* algorithm and propose a hybrid planning algorithm—New Potential Quick-RRT* (NPQ-RRT*), which takes the attitude adjustment angle of the robot into consideration and adds the DWA local planning algorithm. Moreover, the algorithm is extended and applied to the path planning problem of multirobot.

The remainder of the paper is organized as follows. The problem description is addressed in Section 2, and the traditional PQ-RRT* algorithm is explained in Section 3. Then, the hybrid planning algorithm NPQ-RRT* is presented in Section 4. In Section 5, simulation results are provided to show the effectiveness of the proposed approach. Finally, the paper is concluded in Section 6.

## 2. Problem Description

Throughout the paper, $R$ denotes the set of real numbers, $N$ denotes the set of natural numbers, and $R^d$ denotes the space of real $d$-vectors.

We consider an $n$-robot system, where each robot moves in the region. Let $X_{\text{obs}} \subset X$ be the obstacle area and the unobstructed area $X_{\text{free}} = (X/X_{\text{obs}})$. The start position and the target position of the robot $i$ are $x^i_{\text{init}}$ and $x^i_{\text{goal}}$, respectively.

A trajectory of robot $i$ $(i = 1, 2, \ldots, n)$ is defined as follows: $k_i$: $[0, \tau_i] \longrightarrow X$, where $\tau_i$ is the duration of the trajectory. In addition, $k_i(0) = x^i_{\text{init}}$ and $k_i(\tau_i) = x^i_{\text{goal}}$. The trajectory is obstacle-free if $k_i(t) \in X_{\text{free}}$ for all $t \in [0, \tau_i]$. The cost function $c(\cdot)$ finds the path length in terms of Euclidean distance function.

Trajectory $k_i$ is said to be conflict-free if it is obstacle-free and also keeps robot $i$ at a safety distance $d_s > 0$ from all other robots. $\|x_i(t) - x_j(t)\| > d_s$, where $x_i(t)$ and $x_j(t)$ represent the positions of robots $i$ and $j$, $t \in [0, \tau]$, $i, j = 1, 2, \ldots, n$, $i \neq j$, and $\tau = \min(\tau_i, \tau_j)$. The total cost of the trajectories is defined as $\sum_{i=1}^{n} c(k_i)$.

Our goal is to find the trajectory set $K = \{k_1, k_2, \ldots, k_n\}$ in which $k_i$ is conflict-free.

## 3. Related Work

The Rapidly Exploring Random Trees is a sampling-based planning method that builds an undirected graph on a known map through sampling and then finds a relatively optimal path through a search method. The PQ-RRT* is an improved version after adding the target attraction function RGD and the deep parent node search function ***Ancestry***, which can generate a better initial solution and quickly converge to obtain the optimal solution. The pseudocode of the specific algorithm flow is shown in Algorithm 1 [25], where $G = (V, E)$ represents the generated graph.

The related functions are defined as follows:

*SampleFree*: randomly pick points in the global map. Here, the return value is random point $x_{\text{rand}}$.

*RGD*: the adjustment function that adjusts the random point $x_{\text{rand}}$ under the gravitational force of the target point. Here, the return value is the improved sample $x_{\text{prand}}$. *NearestObstacle* function calculates the distance from $x_{\text{prand}}$ to the obstacle space $X_{\text{obs}}$. The parameter $z$ represents the number of iterations. $d_{\text{obs}}$ represents the safety distance, and $\lambda$ represents the step size. The specific pseudocode is shown in Algorithm 2.

*Nearest*: distance evaluation function. The Euclidean distance function is selected in this situation, which returns the node closest to $x_{\text{prand}}$ in the graph $G = (V, E)$ and defines the node as $x_{\text{nearest}}$.

*Steer*: this function connects two given points. The return value of the function is the segment $k$ between the two points.

*CollisionFree*: this function detects whether there is a collision with a static obstacle.

*Near*: given a graph $G = (V, E)$, it returns a set $X_{\text{near}}$, which contains the nodes in the range with $x_{\text{prand}}$ as the center and $r$ as the radius.

*Ancestry*: this function deeply searches the parent node of each point in $X_{\text{near}}$. And it returns the parent node set $X_{\text{sparent}}$ of $X_{\text{near}}$. The specific process is as follows: in

**Input**: $V \longleftarrow x_{\text{init}}$; $E \longleftarrow \varnothing$
**Output**: $G = (V, E)$
(1) **for** $i = 1$ to $n$ **do**
(2) $x_{\text{rand}} \longleftarrow \text{SampleFree}(i)$;
(3) $x_{\text{prand}} \longleftarrow \text{RGD}(x_{\text{rand}})$;
(4) $x_{\text{nearest}} \longleftarrow \text{Nearest}(V, x_{\text{prand}})$;
(5) $k \longleftarrow \text{steer}(x_{\text{nearest}}, x_{\text{prand}})$;
(6) **if** CollisionFree($k$) **then**
(7) $X_{\text{near}} \longleftarrow \text{Near}(V, x_{\text{prand}}, r)$ ;
(8) $X_{\text{sparent}} \longleftarrow \text{Ancestry}(G, X_{\text{near}})$;
(9) $(x_{\text{parent}}, k_{\text{parent}}) \longleftarrow \text{ChooseParent}(X_{\text{near}} \bigcup X_{\text{sparent}}, x_{\text{nearest}}, k)$;
(10) $V \longleftarrow V \bigcup \{x_{\text{prand}}\}$;
(11) $E \longleftarrow E \bigcup \{x_{\text{parent}}, x_{\text{prand}}\}$;
(12) $G \longleftarrow \text{Rewire} - \text{PQ} - \text{RRT} * (G, x_{\text{prand}}, X_{\text{near}})$;
(13) **end**
(14) **end**

Algorithm 1: PQ-RRT* [25].

**Input:** random point $x_{\text{rand}}$, target point $x_{\text{goal}}$
**Output:** improved point $x_{\text{prand}}$
(1) **for** $n = 1$ *to* $z$ **do**
(2) $\overrightarrow{F} = (x_{\text{goal}} - x_{\text{rand}})$
(3) $d_{\min} \longleftarrow \text{NearestObstacle}(X_{\text{obs}}, x_{\text{rand}})$;
(4) **if** $d_{\min} \le d_{\text{obs}}$ **then**
(5) **return** $x_{\text{prand}}$;
(6) **else**
(7) $x_{\text{prand}} \longleftarrow x_{\text{rand}} + \lambda(\overrightarrow{F}/|\overrightarrow{F}|)$;
(8) **end**
(9) **end**

Algorithm 2: RGD($x_{\text{rand}}$).

a given graph $G = (V, E)$, for a node $V_1$, a natural number $p \in N$, if the depth $p = 0$, it returns $\varnothing$ and otherwise returns the $p$th parent of $V_1$.

*ChooseParent*: compare the cost of each path and then determine the parent node $x_{\text{parent}}$ and the path $k_{\text{parent}}$.

*Rewire-PQ-RRT**: a function to generate the final path diagram.

## 4. An Improved Algorithm NPQ-RRT* for Multirobot

*4.1. Overall Ideas.* For the robot formation, we divide it into a leader and several followers. When planning the formation path, we first select the leader as the research object and generate a global path through the global planning algorithm. The path is taken as the target path of the robot formation. Each node in the global path is taken as the local starting point and the local target point. Subsequently, for the movement between the local starting point and the local target point of the robot formation, a local path is generated by the local planning algorithm. Compared with the traditional RRT, RRT*, and other algorithms, the global planning algorithm PQ-RRT* has excellent global search capabilities, but it still has limitations: all these algorithms mentioned above discuss fast random expansion search while ignoring the characteristics of the robot to find a feasible path in the global state. However, when applying this algorithm to practical path planning, the attitude adjustment angle of the robot will have an impact on the operation of the algorithm, as shown in Figure 1.

According to the steps in the traditional RRT* series algorithms, the closest point to Position 1 is evaluated according to the Euclidean distance. It is concluded that Position 2 is the closest point to Position 1. However, for a mobile robot, there is an attitude adjustment angle $\beta$ to Position 2. First, the robot needs to adjust the direction that it faces and then goes to Position 2. For Position 3, it only needs to travel along a straight line. Therefore, the attitude adjustment angle needs to be incorporated into the process of calculating the closest point to Position 1 so as to balance the problems of algorithm convergence time and path smoothness caused by the cumulative rotation angle in the practical application.

At the same time, there are dynamic obstacles in the real environment, which increases the safety risk of mobile robots that perform path planning. The local planning algorithms can better solve these problems and optimize the local path when taking into account dynamic obstacle

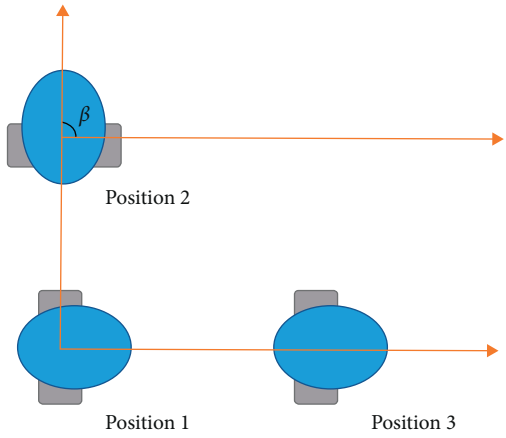


Figure 1: The influence of attitude adjustment angle on the selection of the nearest node.

avoidance. Based on the abovementioned requirements, this paper proposes a hybrid path planning algorithm that considers the attitude adjustment angle.

The local path planner generates a local path for each robot in the formation to follow the local target and avoid obstacles on the local map. By using local targets, the local path planner and the global path planner are combined. The global planner plans the path to the target point in a relatively long period of time for the leader, and the followers follow the leader's path. The local planner updates the trajectory in real time to avoid dynamic and static obstacles. The main idea is shown in Figure 2.

### 4.2. Specific Steps

*4.2.1. The Generation of Global Target Path.* The hybrid planning algorithm NPQ-RRT* proposed in this paper improves the problems of the attitude adjustment and dynamic obstacle avoidance. The leader generates the formation target path through the global planning part of the algorithm.

After applying the new distance evaluation function and local programming algorithm to the PQ-RRT* algorithm, the pseudocode of NPQ-RRT* is shown in Algorithm 3. In the pseudocode, most of the function operations are consistent with the operations of related functions in Algorithm 1. The specific operations of the newly proposed distance evaluation function *NewNearest* are given as follows.

The function incorporates the attitude adjustment angle as a new influencing factor into the distance consideration range:

$$q = \varepsilon_v l + \zeta_w \varphi, \tag{1}$$

where $\varepsilon_v$ and $\zeta_w$ are the evaluation weights of velocity and angular velocity, respectively. Since the local starting point is close to the local target point, $l$ is the Euclidean distance between the $n$th node and the sample point $x_{\text{prand}}$. $\varphi$ is the attitude adjustment angle, as shown in Figure 3.

$l$ and $\varphi$ are defined as follows:

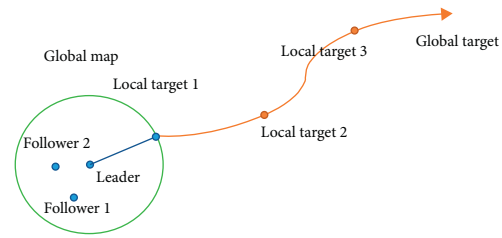


Figure 2: The intersection of global path planner and local path planner.

$$l = \sqrt{(x_v - x_r)^2 + (y_v - y_r)^2},$$
$$\varphi = \phi - \frac{180^\circ \cdot \arctan\left((y_r - y_v / x_r - x_v)\right)}{\pi}, \tag{2}$$

where $(x_v, y_v)$ and $(x_r, y_r)$ are the coordinates of the node and the modified sample $x_{\text{prand}}$. $\phi$ is the angle between the orientation of the mobile robot and the $X$-axis of the global coordinate system, as shown in Figure 3.

Since $l$ and $\phi$ have different units in the new distance evaluation function, it is of no practical significance to directly add and subtract numerically. The selection of the evaluation weights $\varepsilon_v$ and $\zeta_w$ becomes the core of the evaluation function. Under the circumstances, the time to reach the destination can be used to measure the length of the distance. When measuring the distance between two points, we assume that the forward speed and the attitude adjustment angular velocity are constant. Therefore, the new distance evaluation function will select the time as the evaluation index. The smaller the value of the function, the shorter the distance between the two points. The evaluation weights $\varepsilon_v$ and $\zeta_w$ are numerically equal to the reciprocal of the robot's speed and angular velocity at the current moment. Therefore, the new distance evaluation function returns the node that minimizes the function value in the graph $G = (V, E)$ and then defines this node as $x_{\text{new\_nearest}}$.

After the leader completes the steps in the pseudocode, the graph $G = (V, E)$ is obtained which is regarded as the global goal path of the formation.

*4.2.2. The Generation of Local Path.* For each robot in the formation, take the selected $x_{\text{parent}}$ and $x_{\text{prand}}$ as the starting point and target point of the local planning, respectively. Using the function *Local* to obtain the LocalPath, the specific operations are given as follows:

Step 1: generate velocity space [26]. Define the $i$th robot's velocity set $V_{mi}$ as follows:

$$V_{mi} = \left\{ \begin{array}{l} v_i \in [v_{\min}, v_{\max}], \\ \omega_i \in [\omega_{\min}, \omega_{\max}] \end{array} \right\}, \tag{3}$$

where $v_{\min}$ and $v_{\max}$ represent the minimum and maximum velocities that the robot can reach, respectively. $\omega_{\min}$ and $\omega_{\max}$ represent the minimum and maximum angular velocities that the robot can reach, respectively.

**Input:** $V \longleftarrow x_{\text{init}}; E \longleftarrow \varnothing$
**Output:** $G = (V, E)$
(1) **for** $i = 1$ *to* $n$ **do**
(2) $x_{\text{rand}} \longleftarrow \text{SampleFree}(i)$;
(3) $x_{\text{prand}} \longleftarrow \text{RGD}(x_{\text{rand}})$;
(4) $x_{\text{nearest}} \longleftarrow \text{NewNearest}(V, x_{\text{prand}})$;
(5) $k \longleftarrow \text{steer}(x_{\text{new\_nearest}}, x_{\text{prand}})$;
(6) **if** *CollisionFree(k)* **then**
(7) $X_{\text{near}} \longleftarrow \text{Near}(V, x_{\text{prand}}, r)$;
(8) $X_{\text{sparent}} \longleftarrow \text{Ancestry}(G, X_{\text{near}})$;
(9) $(x_{\text{parent}}, k_{\text{parent}}) \longleftarrow \text{ChooseParent}(X_{\text{near}} \bigcup X_{\text{sparent}}, x_{\text{nearest}}, k)$
(10) $\text{LocalPath} \longleftarrow \text{Local}(x_{\text{parent}}, x_{\text{prand}})$
(11) $V \longleftarrow V \bigcup \{x_{\text{prand}}\}$;
(12) $E \longleftarrow E \bigcup \{x_{\text{parent}}, x_{\text{prand}}\}$;
(13) $G \longleftarrow \text{Rewire} - \text{NPQ} - \text{RRT} * (G, x_{\text{prand}}, X_{\text{near}})$;
(14) **end**
(15) **end**

Algorithm 3: NPQ-RRT*.

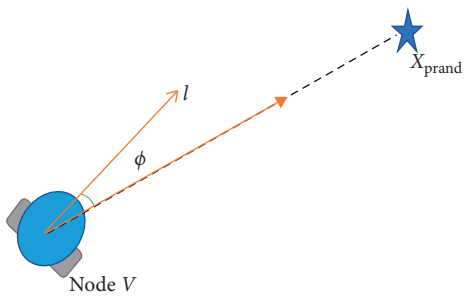


Figure 3: The attitude adjustment of the mobile robot.

Due to the limited torque of the motor, there are maximum acceleration and deceleration limits. The achievable velocity set $V_{di}$ affected by the motor performance is defined as

$$V_{di} = \{(v_i, \omega_i) | v_i \in [v_{ci} - a_b \Delta t, v_{ci} + a_m \Delta t] \cap \omega_i \in \cdot [\omega_{ci} - \alpha_b \Delta t, \omega_{ci} + \alpha_m \Delta t]\}, \quad (4)$$

where $v_{ci}$ and $\omega_{ci}$ are the current velocity and angular velocity of the $i$th mobile robot, respectively. $a_b$ and $a_m$ correspond to the maximum acceleration during deceleration and the maximum acceleration during acceleration. $\alpha_b$ and $\alpha_m$ correspond to the maximum angular acceleration during deceleration and the maximum angular acceleration during acceleration. The opposite direction of the original movement direction is defined as the deceleration direction.

When the robot decelerates with the maximum acceleration at the current velocity, it can be guaranteed to stop before encountering an obstacle, then the velocity is safe. The safe velocity set $V_{si}$ is defined as follows:

$$V_{si} = \left\{(v_i, \omega_i) | v_i \le \sqrt{2 \cdot d(v_i, \omega_i) \cdot a_b} \cap \omega_i \le \sqrt{2 \cdot d(v_i, \omega_i) \cdot \alpha_b}\right\}, \quad (5)$$

where the function $d(v_i, \omega_i)$ represents the distance between the $i$th robot and the nearest obstacle on the current trajectory.

The final definition of the $i$th robot's feasible velocity space set is

$$V_{ai} = V_{mi} \cup V_{di} \cup V_{si}. \quad (6)$$

Step 2: obtain the predicted trajectory corresponding to each velocity and avoid collisions within the formation.

First of all, we need to build a model of the robot. Assume that the robot only has two movement modes: forward and rotating. At the current moment, the robot has velocities $v(t)$ and $\omega(t)$. Consider two adjacent moments, as shown in Figure 4. Since the robot's adjacent moment $\Delta t$ (usually measured by the code disc sampling period in ms) is relatively short, the motion at the two adjacent moments can be regarded as uniform motion. The trajectory of the motion can be regarded as a straight line. Within $\Delta t$, the robot moves $v(t)\Delta t$ in the current direction. $\phi(t)$ is the angle between the direction of the mobile robot and the $X$-axis of the global coordinate system.

In the real coordinate system, the displacement $\Delta x$ of the robot moving in the $X$-axis direction of the global coordinate system and the displacement $\Delta y$ of the $Y$-axis movement in the global coordinate system are defined as follows:

$$\Delta x = v(t)\Delta t \cos\phi(t), \quad \Delta y = v(t)\Delta t \sin\phi(t). \quad (7)$$

As for the trajectory of the robot, $t$ represents the previous moment and $t+1$ represents the current moment. $x(t+1)$, $y(t+1)$, and $\phi(t+1)$ represent the robot's position information and orientation angle information, respectively, which are defined as

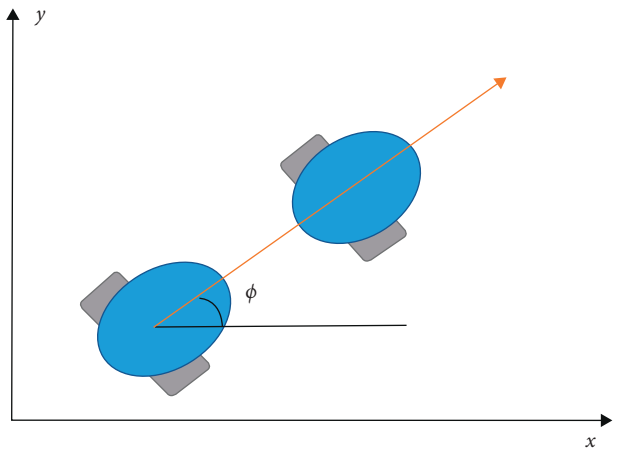


Figure 4: The model of the mobile robot.

$$x(t+1) = x(t) + v(t)\Delta t\cos\phi(t), \tag{8}$$

$$y(t+1) = y(t) + v(t)\Delta t\sin\phi(t), \tag{9}$$

$$\phi(t+1) = \phi(t) + \omega(t)\Delta t. \tag{10}$$

Substituting the velocity space $V_{ai}$ of the robot obtained in the first step into equations (8)–(10), we can obtain the corresponding trajectory expression.

In addition, due to the possibility of collisions between the robots in the robot formation, we make further constraints. For the obtained local predicted trajectories of two adjacent robots, when the predicted trajectories of the two robots have no intersection, they will not collide. On the contrary, when the two predicted trajectories have intersections, they may collide, as shown in Figure 5.

Define the positions of the two robots as $p_i(t)$ and $p_j(t)$, respectively. The meeting point is $m(t)$ and the velocities of the two robots are $v_i(t)$ and $v_j(t)$. The distances $\Delta l_1$ and $\Delta l_2$ are defined as

$$\begin{aligned} \Delta l_1 &= \left\| p_i(t) - m(t) \right\|, \\ \Delta l_2 &= \left\| p_j(t) - m(t) \right\|. \end{aligned} \tag{11}$$

According to the distance, we can constrain the velocities of the two robots: if $\Delta l_1 \geq \Delta l_2$, then

$$\begin{aligned} v_i(t+1) &= v_i(t) - a_b \cdot \epsilon, \\ v_j(t+1) &= v_j(t) + a_m \cdot \epsilon. \end{aligned} \tag{12}$$

Else, if $\Delta l_1 \leq \Delta l_2$, then

$$\begin{aligned} v_i(t+1) &= v_i(t) + a_m \cdot \epsilon, \\ v_j(t+1) &= v_j(t) - a_b \cdot \epsilon, \end{aligned} \tag{13}$$

where $\epsilon$ is a constant that can be set according to the relationship between the velocity and the maximum acceleration.

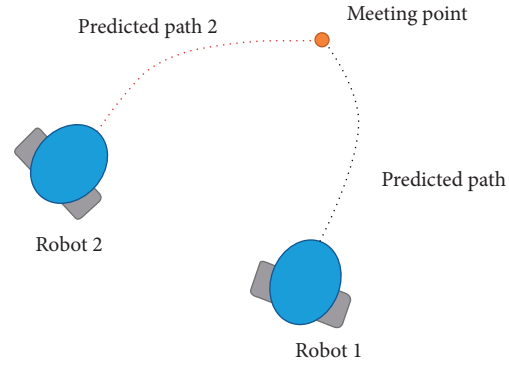


Figure 5: The model of multirobots.

Step 3: select the locally optimal path through the evaluation function. Define the evaluation function of the $i$th robot's local path as

$$\begin{aligned} G(v_i, \omega_i) = {} & \alpha \cdot m\_\text{head}(v_i, \omega_i) + \delta \cdot m\_d(v_i, \omega_i) \\ & + \gamma \cdot m\_\text{vel}(v_i, \omega_i) - \rho \cdot m\_\text{meet}(v_i, \omega_i), \end{aligned} \tag{14}$$

where $(v_i, \omega_i) \in V_{ai}$ and the variables $\alpha, \delta, \gamma$ and $\rho$ are the initial weights of the function. $m$ is the total number of all trajectories sampled. The function $m\_\text{head}(v_i, \omega_i)$ is used to evaluate the heading score, which is defined as follows:

$$m\_\text{head}(v_i, \omega_i) = \frac{\text{head}(v_i, \omega_i)}{\sum_{c=1}^{m} \text{head}(v_i, \omega_i)}. \tag{15}$$

The function $\text{head}(v_i, \omega_i)$ is defined as

$$\text{head}(v_i, \omega_i) = 180^\circ - \theta_i, \tag{16}$$

where $\theta_i$ is the angle between the current direction that the mobile robot is facing and the direction when it reaches the local target point. It can be defined as

$$\theta_i = \phi - \frac{180^\circ \cdot \arctan\left(\left(y_b - y_p / x_b - x_p\right)\right)}{\pi}, \tag{17}$$

where $(x_b, y_b)$ is the coordinate of the local target point in the global map and $(x_p, y_p)$ is the coordinate of the predicted position of the $i$th robot in the global map.

The function $m\_d(v_i, \omega_i)$ is used to evaluate the safety distance score, which is defined as follows:

$$m\_d(v_i, \omega_i) = \frac{d(v_i, \omega_i)}{\sum_{c=1}^{m} d(v_i, \omega_i)}. \tag{18}$$

The function $d(v_i, \omega_i)$ can be defined as

$$d(v_i, \omega_i) = \min\left\{\text{dist}\left(x_i, s_j\right)\right\}, \tag{19}$$

where $x_i$ is a random point on the trajectory, $s_j$ represents the corresponding obstacle, and *dist* is a function to calculate Euclidean distance. If there are no obstacles on this trajectory, set $d(v_i, \omega_i)$ as a constant.

The function $m\_\text{vel}(v_i, \omega_i)$ is used to evaluate the speed score of the current trajectory. The function $\text{vel}(v_i, \omega_i)$

corresponds to the velocity value of the current trajectory. $m\_\mathrm{vel}(v_i, \omega_i)$ is defined as follows:

$$m\_\mathrm{vel}(v_i, \omega_i) = \frac{\mathrm{vel}(v_i, \omega_i)}{\sum_{c=1}^{m} \mathrm{vel}(v_i, \omega_i)}. \quad (20)$$

The function $m\_\mathrm{meet}(v_i, \omega_i)$ is used to evaluate the score of internal collision probability. The function $\mathrm{meet}(v_i, \omega_i)$ corresponds to the number of intersections with the predicted trajectories of other robots in the formation. $m\_\mathrm{meet}(v_i, \omega_i)$ is defined as follows:

$$m\_\mathrm{meet}(v_i, \omega_i) = \frac{\mathrm{meet}(v_i, \omega_i)}{\sum_{c=1}^{m} \mathrm{meet}(v_i, \omega_i)}. \quad (21)$$

The path with the highest overall score obtained by the evaluation function evaluation is the optimal path. This path is regarded as the local path and is combined with the global planning path to obtain the final path.

## 5. Simulation

In this section, we discuss the practical effects of the proposed hybrid path planning algorithm NPQ-RRT* in complex environments with dynamic and static obstacles. We perform a comparative simulation experiment through MATLAB on an Intel Core i5 4-core, 8 GB RAM computer.

*5.1. Build a Simulation Environment and Set Relevant Parameters.* We first construct a $1000 \times 1000$ map environment. We set (0,0) as the starting point for global planning and (1000,1000) as the target point for global planning. There are 5 static obstacles, which are represented by green rectangles. The obstacles are distributed in this environment, as shown in Figure 6.

After the command to start path planning is issued, additional obstacles (not overlapping with the current mobile robot position) will be randomly generated as dynamic obstacles in the environment at any given time, as shown in Figure 7.

In this simulation, the relevant parameters of the NPQ-RRT* hybrid path planning algorithm are set as follows: in the *RGD* function, $\lambda = d_{\mathrm{obs}} = 1$ and $z = 80$. In the *ChooseParent*function, $d = 2$. In the *Local* function, the initial weights $\alpha, \delta, \gamma$, and $\rho$ of the evaluation function are set to 0.05, 0.2, 0.1, and 0.1, respectively. The maximum speed of the mobile robot is 1 m/s, the maximum angular speed is 0.5 rad/s, and the acceleration range is $[-1, 1]$. The angular acceleration range is $[-0.5, 0.5]$, the velocity resolution is 5 m/s, and the angular velocity resolution is 6 rad/s.

*5.2. The Simulation Design and Results*

*5.2.1. Group 1.* To verify the performance of using the improved distance evaluation function in the path planning process of mobile robots, we conduct 10 sets of comparative tests without considering dynamic obstacles. The original PQ-RRT* algorithm and our proposed NPQ-RRT* perform the same path planning missions. The sum of the changes in the heading angle between the nodes makes up the entire adjustment angles. The adjustment angular velocity $\omega = 0.5$ rad/s is used for obtaining the posture adjustment time. After conducting 10 groups of calculations, the average results in Table 1 are obtained.

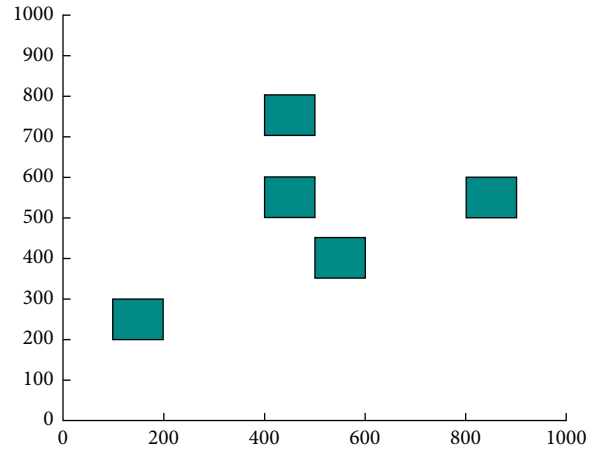


Figure 6: The global environment of simulation.

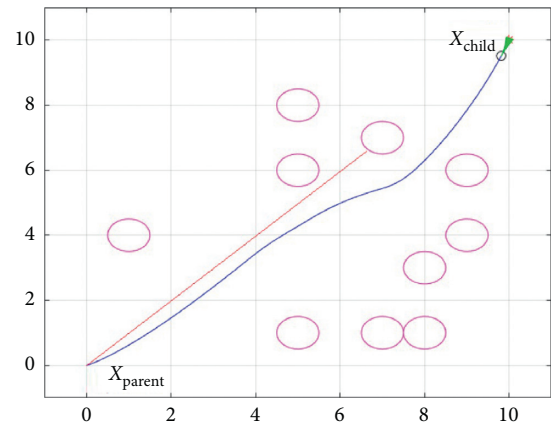


Figure 7: The intersection of global path planner and local path planner.

Figures 8 and 9 show the results of path planning for NPQ-RRT* and PQ-RRT*, respectively. It can be seen that the improved algorithm's robot attitude adjustment time is shortened, the algorithm's running speed is sharply accelerated, and the resulting path is smoother than the original PQ-RRT*.

*5.2.2. Group 2.* To verify that the hybrid path planning algorithm NPQ-RRT* after adding local planning has better dynamic obstacle avoidance performance than the original PQ-RRT* algorithm, we carry out the following comparative test in consideration of obstacle dynamic obstacles, as shown in Figure 7.

The confirmed $x_{\mathrm{prand}}$ and $x_{\mathrm{parent}}$ are used as the target point and starting point of local planning. The original algorithm PQ-RRT* lacks a local path planning algorithm; that is, it moves along the line of $x_{\mathrm{child}}$ and $x_{\mathrm{parent}}$

Table 1: Comparing the time of attitude adjustment.

| Number | PQ-RRT* | NPQ-RRT* |
|---|---|---|
| 1 | 65.971 | 59.311 |
| 2 | 56.567 | 50.889 |
| 3 | 57.288 | 45.499 |
| 4 | 63.755 | 47.139 |
| 5 | 73.209 | 62.714 |
| 6 | 67.423 | 41.650 |
| 7 | 59.422 | 64.298 |
| 8 | 64.478 | 52.669 |
| 9 | 71.492 | 49.184 |
| 10 | 68.424 | 51.770 |
| Average | 64.703 | 52.512 |

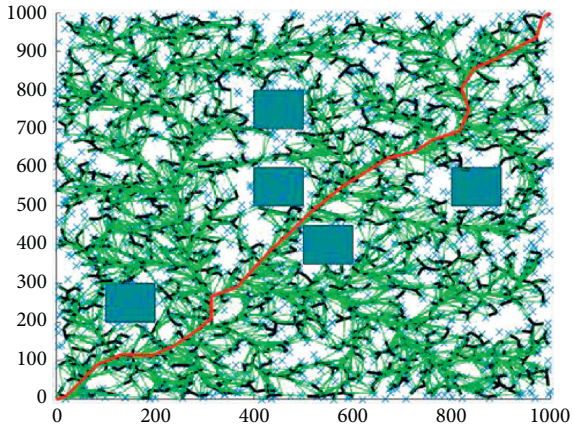


Figure 8: The path planned by NPQ-RRT*.

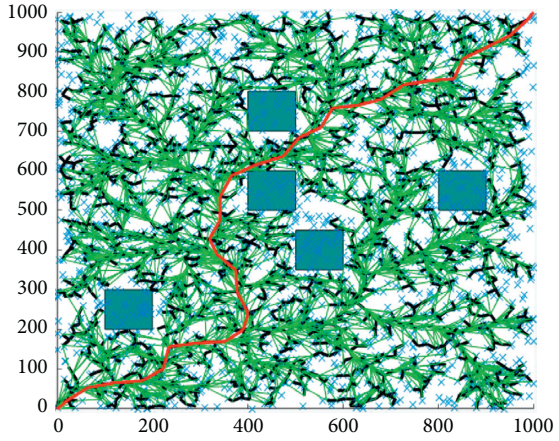


Figure 9: The path planned by PQ-RRT*.

(represented by the red line in Figure 7). It is unable to reach the target due to the existence of dynamic obstacles, causing path planning to fail. After using NPQ-RRT* with local planning algorithm, the mobile robot successfully avoids obstacles and reaches the local target point (the blue line in Figure 7).

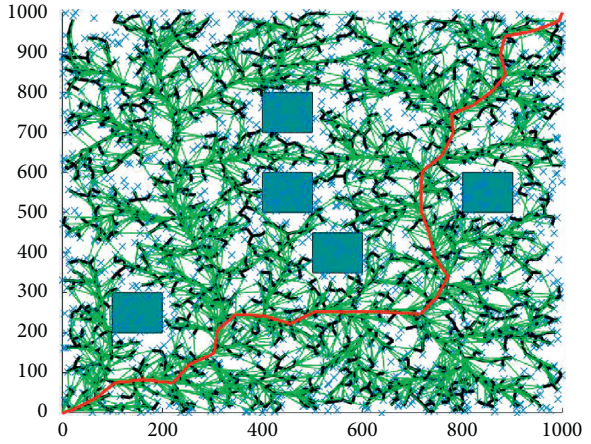


Figure 10: The path planned by hybrid RRT.

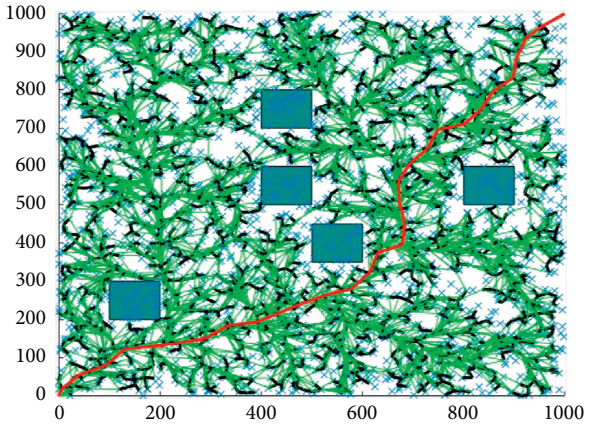


Figure 11: The path planned by NPQ-RRT*.

*5.2.3. Group 3.* To compare the performance of the common RRT mixed planning algorithm and NPQ-RRT*, we perform experiments in the same obstacle environment. The results are obtained, as shown in Figures 10 and 11.

It can be seen that NPQ-RRT* can obtain a smoother path with a shorter overall length and better overall performance compared to the ordinary RRT mixed planning algorithm.

*5.2.4. Group 4.* To test the performance of the algorithm when applied to multirobot path planning, we take three robots as an example to perform the following test. Among them, the small black circle represents the robot. The green area represents the detection range of the robotic lidar. The big red circle represents obstacles randomly generated on the map. The blue lines represent the local paths generated by each robot. We use confirmed $x_{\text{prand}}$ and $x_{\text{parent}}$ as the target point and starting point of the formation planning. Obstacles are randomly generated on the map, and the robot formation moves from the starting point to the target point at the same time. For randomly generated obstacles, the three robots use local planning algorithms to plan their paths

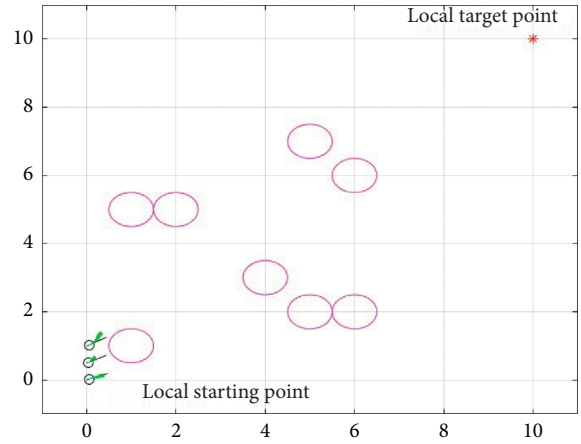


Figure 12: The initial states of the robots.

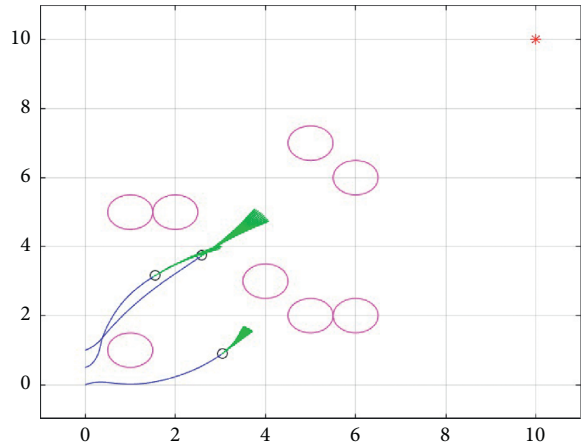

Figure 13: The positions of the robots: $t = 10$s.

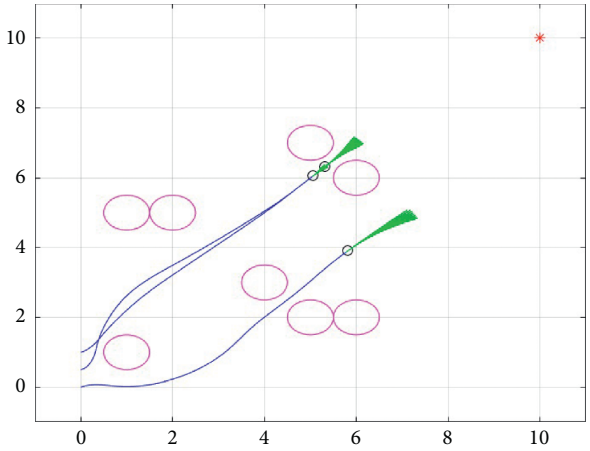

Figure 14: The positions of the robots: $t = 15$s.

to generate obstacle-free paths. At the same time, each robot in the robot formation adopts an avoidance strategy to maintain a safe distance to avoid collisions in the formation. The result is shown in Figures 12–15.

Through the simulation results, we can find that the robot formation completes the obstacle avoidance to the target point, which verifies the feasibility of the proposed algorithm applied to the robot formation.

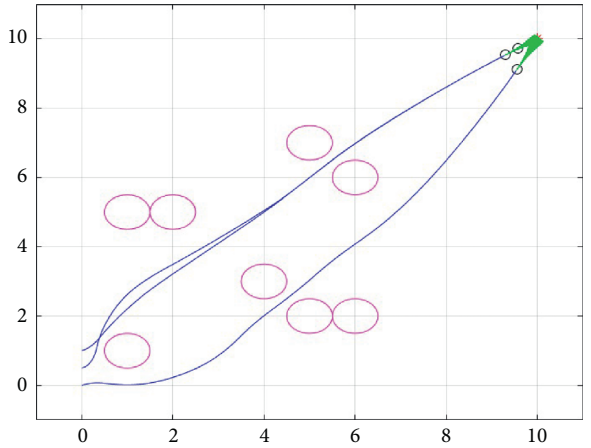

Figure 15: The positions of the robots: $t = 20$s.

Based on the above simulation results, it can be found that NPQ-RRT* has better dynamic obstacle avoidance ability and can effectively shorten the attitude adjustment time and get a smoother path. In addition, this algorithm can also obtain ideal results when it is applied to the robot formation path planning.

## 6. Conclusions

This paper proposes a hybrid path planning algorithm NPQ-RRT*, which studies the path planning of multi-robot in an environment with dynamic and static obstacles. NPQ-RRT* chooses the improved version of the Rapidly Exploring Random Trees algorithm PQ-RRT* as the global planning algorithm. Combined with the attitude adjustment angle of the mobile robot, we propose a new distance evaluation function, which optimizes the selection of the nearest node. After the parent node and the child node are identified, the local planning step is added. The parent node and child node are used as the local starting point and the local target point to generate a local path avoiding dynamic obstacles. The global path is obtained by tracking the local target point. At the same time, the algorithm optimizes the potential collisions within the robot formation to ensure the safety of the robots. Also, the potential collision possibility in the formation is added as a new evaluation index into the evaluation function to select the optimal path. The simulation results show that compared with PQ-RRT*, the hybrid path planning algorithm NPQ-RRT* has better dynamic obstacle avoidance ability. Furthermore, it can get a relatively better path compared with the ordinary RRT hybrid planning algorithm. When applied to the path planning of robot formation, it can effectively shorten the attitude adjustment time and obtain a smoother path.

## Data Availability

No data were used to support this study.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under grant no. 61973064, the Natural Science Foundation of Hebei Province of China under grant no. F2019501126, the Natural Science Foundation of Liaoning Province of China under grant no. 2020-KF-11-03, and the Fundamental Research Funds for the Central Universities under grant no. N182304013.

## References

[1] S. M. M. Rahman, "Cyber-physical-social system between a humanoid robot and a virtual human through a shared platform for adaptive agent ecology," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, 2017.

[2] L. Cheng, T. Yu, X. Zhang, and B. Yang, "Parallel cyber-physical-social systems based smart energy robotic dispatcher and knowledge automation: concepts, architectures and challenges," *IEEE Intelligent Systems*, vol. 34, no. 2, pp. 54–64, 2018.

[3] Z. Zhu, Y. Wen, Z. Zhang, Z. Yan, S. Huang, and X. Xu, "Accurate position estimation of mobile robot based on cyber-physical-social systems (CPSS)," *IEEE Access*, vol. 8, pp. 56359–56370, 2020.

[4] H. Tang, L. Li, and N. Xiao, "Smooth sensor motion planning for robotic cyber physical social sensing (CPSS)," *Sensors*, vol. 17, no. 2, p. 393, 2017.

[5] C. Yang, G. Ganesh, S. Haddadin, S. Parusel, A. Albu-Schaeffer, and E. Burdet, "Human-like adaptation of force and impedance in stable and unstable interactions," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 918–930, 2011.

[6] F. Chen, X. Chen, L. Xiang, and W. Ren, "Distributed economic dispatch via a predictive scheme: heterogeneous delays and privacy preservation," *Automatica*, vol. 123, 2020.

[7] F. Chen and W. Ren, "Sign projected gradient flow: a continuous-time approach to convex optimization with linear equality constraints," *Automatica*, vol. 120, 2020.

[8] Z. Li, P. Y. Tao, S. S. Ge, M. Adams, and W. S. Wijesoma, "Robust adaptive control of cooperating mobile manipulators with relative motion," *IEEE Transactions on Systems Man & Cybernetics Part B*, vol. 39, no. 1, pp. 103–116, 2009.

[9] M. Chen, S. S. Ge, and B. Ren, "Robust attitude control of helicopters with actuator dynamics using neural networks," *IET Control Theory & Applications*, vol. 4, no. 12, pp. 2837–2854, 2010.

[10] Z. Li, X. Cao, and N. Ding, "Adaptive fuzzy control for synchronization of nonlinear teleoperators with stochastic time-varying communication delays," *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 4, pp. 745–757, 2011.

[11] R. Cui, J. Guo, and B. Gao, "Game theory-based negotiation for multiple robots task allocation," *Robotica*, vol. 31, no. 6, pp. 923–934, 2013.

[12] F. Chen and J. Chen, "Minimum-energy distributed consensus control of multiagent systems: a network approximation approach," *IEEE Transactions on Automatic Control*, vol. 65, no. 3, pp. 1144–1159, 2020.

[13] L. Xiang, F. Chen, W. Ren, and G. Chen, "Advances in network controllability," *IEEE Circuits & Systems Magazine*, vol. 19, no. 2, pp. 8–32, 2019.

[14] Y. Huang, Z. Li, Y. Jiang, and L. Cheng, "Cooperative path planning for multiple mobile robots via hafsa and an expansion logic strategy," *Applied Sciences*, vol. 9, no. 4, 2019.

[15] A. Majeed and S. Lee, "A new coverage flight path planning algorithm based on footprint sweep fitting for unmanned aerial vehicle navigation in urban environments," *Applied Sciences*, vol. 9, no. 7, p. 1470, 2019.

[16] H. Y. Lee, H. Shin, and J. Chae, "Path planning for mobile agents using a genetic algorithm with a direction guided factor," *Electronics*, vol. 7, no. 10, 2018.

[17] J. Wang, J. Xu, and R. Tai, "A bi-level probabilistic path planning algorithm for multiple robots with motion uncertainty," *Complexity*, vol. 2020, Article ID 9207324, , 2020.

[18] P. Cui, W. Yan, R. Cui, and J. Yu, "Smooth path planning for robot docking in unknown environment with obstacles," *Complexity*, vol. 2018, Article ID 4359036, 17 pages, 2018.

[19] S. M. LaValle, "Rapidly-exploring random trees: a new tool for path planning," 1998.

[20] J. J. K. Jr and S. M. Lavalle, "RRT-Connect: An efficient approach to single-query path planning," in *Proceedings of the 2000 IEEE International Conference on Robotics and Automation, ICRA 2000*, San Francisco, CA, USA, April 2000.

[21] A. T. Perez, S. Karaman, A. C. Shkolnik, E. Frazzoli, and M. R. Walter, "Asymptotically-optimal path planning for manipulation using incremental sampling-based algorithms," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots & Systems*, Tokyo, Japan, November 2013.

[22] A. H. Qureshi and Y. Ayaz, "Potential functions based sampling heuristic for optimal path planning," *Autonomous Robots*, vol. 40, no. 6, pp. 1079–1093, 2016.

[23] A. Azzabi and K. Nouri, "An advanced potential field method proposed for mobile robot path planning," *Transactions of the Institute of Measurement and Control*, vol. 41, no. 11, pp. 3132–3144, 2019.

[24] I.-B. Jeong, S.-J. Lee, and J.-H. Kim, "Quick-RRT*: triangular inequality-based implementation of RRT* with improved initial solution and convergence rate," *Expert Systems with Applications*, vol. 123, pp. 82–90, 2019.

[25] Y. Li, W. Wei, Y. Gao, D. Wang, and Z. Fan, "PQ-RRT*: an improved path planning algorithm for mobile robots," *Expert Systems with Applications*, vol. 45, Article ID 113425, 2020.

[26] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 2002.

[27] P. Tang, R. Zhang, D. Liu, L. Huang, G. Liu, and T. Deng, "Local reactive obstacle avoidance approach for high-speed unmanned surface vehicle," *Ocean Engineering*, vol. 106, pp. 128–140, 2015.

[28] H. Yang, J. Qi, Y. Miao, H. Sun, and J. Li, "A new robot navigation algorithm based on a double-layer ant algorithm and trajectory optimization," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 11, pp. 8557–8566, 2019.

[29] Z. Chen, Y. Zhang, Y. Zhang, Y. Nie, J. Tang, and S. Zhu, "A hybrid path planning algorithm for unmanned surface vehicles in complex environment with dynamic obstacles," *IEEE Access*, vol. 7, pp. 126439–126449, 2019.

[30] G. Huskić, S. Buck, and A. Zell, "Gerona: generic robot navigation," *Journal of Intelligent & Robotic Systems*, vol. 95, no. 2, 2018.