IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 8, 2019, accepted December 6, 2019, date of publication December 10, 2019, date of current version January 3, 2020.

Digital Object Identifier 10.1109/ACCESS.2019.2958876

A Heuristic Rapidly-Exploring Random Trees Method for Manipulator Motion Planning

CHENGREN YUAN[1], WENQUN ZHANG[1], GUIFENG LIU[2], XINGLONG PAN[1], AND XIAOHU LIU[3]

[1]Power Engineering Department, Naval University of Engineering, Wuhan 430033, China
[2]Office of Educational Administration, Naval University of Engineering, Wuhan 430033, China
[3]College of Electrical Engineering, Naval University of Engineering, Wuhan 430033, China

Corresponding author: Guifeng Liu (497539764@qq.com)

This work was supported in part by the National Natural Science Foundation of China under Grant NSFC 51509255, and in part by the Nava Engineering University Independent Research under Grant 425317Q033.

ABSTRACT In order to plan the robot path in 3D space efficiently, a modified Rapidly-exploring Random Trees based on heuristic probability bias-goal (PBG-RRT) is proposed. The algorithm combines heuristic probabilistic and bias-goal factor, which can get convergence quickly and avoid falling into a local minimum. Firstly, PBG-RRT is used to plan a path. After obtaining path points, path points are rarefied by the Douglas-Peucker algorithm while maintaining the original path characteristics. Then, a smooth trajectory suitable for the manipulator end effector is generated by Non-uniform B-spline interpolation. Finally, the effector is moving along the trajectory by inverse kinematics solving angle of joint. The above is a set of motion planning for the manipulator. Generally, 3D space obstacle avoidance simulation experiments show that the search efficiency of PBG-RRT is increased by 217%, while search time is dropped by 168% compared with P-RRT (Heuristic Probability RRT). After rarefying, the situation where the path oscillated around the obstacle is corrected effectively. And a smooth trajectory is fitted by spline interpolation. Ultimately, PBG-RRT is verified on the ROS (Robot Operating System) with the Robot-Anno manipulator. The results reveal that the validity and reliability of PBG-RRT are proofed in obstacle avoidance planning.

INDEX TERMS Bias-target factor, manipulator, motion planning, rapidly-exploring random trees, rarefy, heuristic probability, ROS, non-uniform B-spline.

I. INTRODUCTION

Motion planning is divided into path planning and trajectory planning [1]–[3]. Path planning focuses on generating a path, and trajectory planning gives information of time to a path [4]. Research on the manipulator end-effector, the main problem of path tracing is studied in path planning. Moreover, in trajectory planning, the practical application of kinematics inverse solution (inverse attitude, velocity and acceleration inverse solution) is realized according to the requirements of the task [5], [6]. In view of modern development, the manipulator is widely used in modern manufacturing and education industry, which are shown in Figure 1. Therefore, systematic motion planning is important for solving practical problems.

The Rapidly-exploring Random Trees (RRT)algorithm is a fast algorithm of path planning based on random sampling. It is widely used by researchers because RRT can search in high-dimensional space and avoid the difficulty of 3D modeling effectively [7]–[9]. Due to the randomness and blindness of RRT [6], many scholars have proposed improved RRT algorithms for different research objects and performance requirements. Among them, LaValle introduced

(a) (b)

FIGURE 1. The application of the manipulator in teaching and industry. (a) The manipulator has been used for teaching and studying in the university laboratory. (b) The manipulator has become an important component of the assembly line in the industry.

The associate editor coordinating the review of this manuscript and approving it for publication was Liang Hu.

probability-based sampling, which improved the search efficiency of the basic RRT algorithm. But at the same time, it was easy to fall into the local minimum [10]. Kuffner and LaValle proposed the RRT-Connect algorithm to increase the path generation speed by increasing the number of extensional trees [11]. Boyaciogl proposed time of optimal RRT algorithm that combined the actual path and verified the feasibility [12]. Karaman proposed the path optimized RRT algorithm, which had high stability and neared optimally, but it took a lot of time [13]. Cao X employed the idea of target gravity to search path and used the genetic algorithm (GA) to smooth processing. The optimized path length was shortened [14]. Zhang H introduced a regression mechanism to prevent over-searching configuration space, adopted an adaptive expansion mechanism. To some extent, the path was improved efficiently [15]. Xinyu W integrated the bidirectional artificial potential field into the rapidly exploring random tree star (RRT$*$). There was a significant improvement in search time and the number of iterations in path planning [16]. Moon C proposed dual-tree rapidly exploring random tree (DT-RRT), which showed great advantages on time and success rate. It also generated a quality trajectory without considering robot kinematics [17]. Wei K proposed a path based on the maximum curvature constraint and achieved the smoothness of the generation path better. However, there were certain limitations in the dynamic obstacle avoidance tracking process [7].

Path planning and manipulator motion evaluation indicators are combined in the abovementioned studies. Furthermore, these methods are made based on the RRT algorithm or RRT-derived algorithm, and some satisfactory results are obtained. Overall, the problems are divided into the following five parts: (1) The local minimum situation, which is the main reason for failure in the search process, cannot be solved well. (2) Similarly, the phenomenon of oscillatory motion near obstacle increases in planning time. That means the algorithm is unreliable for motion planning. (3) In otherwise, motion planning is separated in most studies. Hardly studies have proposed a complete solution to solve the manipulator motion planning problem. (4) In researches, complementarity is rarely considered in algorithms, which can help to improve the reliability of the motion planning system. So, the disadvantage caused by a single algorithm is averted. For example, the path point of PBG-RRT is denseness at the beginning but is sparseness when approaching the goal. According to the case, the Non-Uniform B-Spline is used for fitting, and the manipulator trajectory can be obtained more accurately. (5) The most of manipulator motion planning is simulated in the 2D environment. Comparing 2D environment simulation, the 3D environment can reflect the feasibility of the algorithm more directly. At the same time, other path planning problems can be applied in 3D space for PBG-RRT, especially in Unmanned Aerial Vehicle problems.

In this paper, an improved RRT path planning with robust performance in complex environments. Foremostly, combined with the heuristic probability and bias-goal factor, the PBG-RRT algorithm is proposed. which can avert the local minimum and time-consuming situations. After that, the Douglas-Peucker algorithm is adopted for generating path points to cut down the control point which is reversed by the path point. After rarefying the path points, a trajectory curve is fitted by Non-Uniform B-Spline. Finally, the validity and reliability of the whole motion planning strategy are verified on ROS. Also, the feasibility and advantages of the PBG-RRT algorithm are verified.

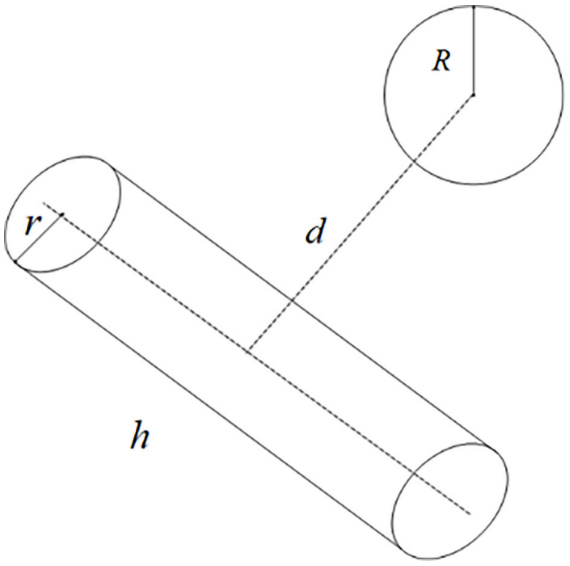


FIGURE 2. Simplified collision detection diagram. The cylinder is a simplification of the mechanical arm in the configuration space, the radius is r. The sphere is a simplification of the obstacle in the configuration space, the radius is R. The distance between the sphere center and the center axial is d.

The remainder of this paper is as follows. Section II, collision detection is introduced base on the problems with maps. Then the improved RRT algorithm (PBG-RRT) is introduced in section III. The trajectory is fitted by Non-Uniform B-Spline after rarefying path points in section IV. In Section V, simulation and verification are carried out used by Matlab and ROS to verify the whole motion planning strategy. Finally, a conclusion is summarized in section VI.

II. COLLISION DETECTION AND MAP SETTING

A. COLLISION DETECTION PROBLEMS AND SOLUTIONS

Collision detection usually simplifies the obstacle and manipulator model by geometric envelope method in space [18]. As shown in Figure 2, set the coordinate of the sphere center (x, y, z). The distance from the sphere center to the center axial of the cylinder is d. And the sphere radius is R. Also, the radius of the cylinder is r. Thus, a simplified collision problem converts into when $d > r + R$ the manipulator does not collide with an obstacle, otherwise, it collides. Although this approach takes up a certain amount of effective space, calculation efficiency is improved.

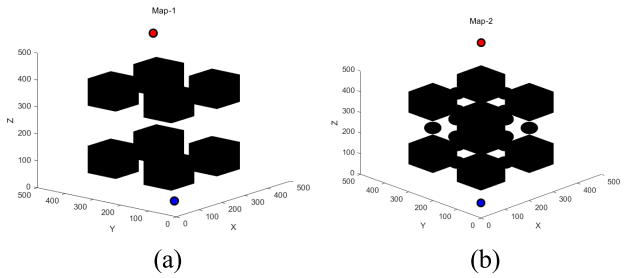


FIGURE 3. The simulation maps in Matlab. (a) Map-1 is used to compare search capabilities with different algorithms. (b) Map-2 is used to verify the PBG-RRT tracing capabilities in complex space. Coordinate space is set as 500 × 500 × 500. The blue point is the start. The red point is the goal. Black geometry is an obstacle.

B. MAPS SETTING

Maps are established according to the comparison and verification with PBG-RRT, which is shown in Figure 3. Map-1 is used to prove the validity of the algorithm. And Map-2 is used to verify the PBG-RRT algorithm reliability in complex space.

III. IMPROVED RRT PATH PLANNING

A. THEORY OF RRT ALGORITHM

RRT algorithm is traversing the whole graph by probability. And it facilitates searching in high-dimensional space [19], [20]. Figure 4 shows the growth process of RRT.

According to Algorithm 1, set a start point q_{start} and store q_{start} in q_{nodes}. The random sampling point of the whole map in space is q_{rand}. Find the nearest point in q_{nodes} to q_{rand} as q_{near}. Then, advance to q_{new} in a certain step δ in the direction of q_{near} to q_{rand}. Collision detection is executed during this process. If the collision is not detected, store q_{new} in q_{nodes}. If the collision is detected, it will re-sampling to repeat the above process. When $|q_{new} - q_{goal}| < E\text{rror}$ it is considering finding a goal and Store q_{goal} in q_{nodes}. Finally, according to the parent-child relationship of each node in q_{nodes}, the reverse search finds the planned path [21], [22]. Algorithm 1 presents the pseudo-code of RRT. There are four return values in the algorithm: "Advanced" means that a new node is searched but it is not known whether it is near the goal

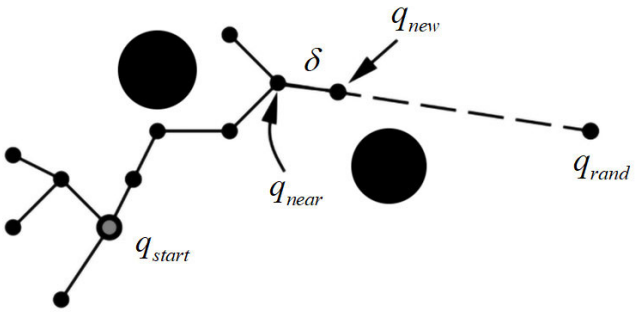


FIGURE 4. RRT growth process diagram. Black circles are obstacles. Black points are RRT path points. δ is the step of the RRT tree movement. q_{start} is the beginning of the RRT tree. q_{rand} is a random point based on spatial sampling. q_{near} is the closest point in the RRT tree to q_{rand}. q_{new} is a new path point that RRT tree in δ step size along with the random point.

Algorithm 1 Base RRT

Input:
Initial configuration q_{start} q_{goal} *obstacle*
Number of sampling point K
Step size δ

Output:
RRT graph G
Vertices of tree *nodes*

Initialize all Parameters ;
$nodes = q_{start}$;
for $i = 1$ *to* K **do**
  q_{rand} = Sample();
  $q_{near} \leftarrow$ Nearest($nodes$, q_{rand});
  $q_{new} \leftarrow$ Steer(q_{near}, q_{rand}, δ);
  if *ObstacleFree*(q_{near}, q_{new}) **then**
    $nodes$.add(q_{new});
    return Advanced;
if $dist(q_{new} - q_{goal}) < Error$ **then**
  return Reached;
else
  return Trapped
final ;
return *Graph*;

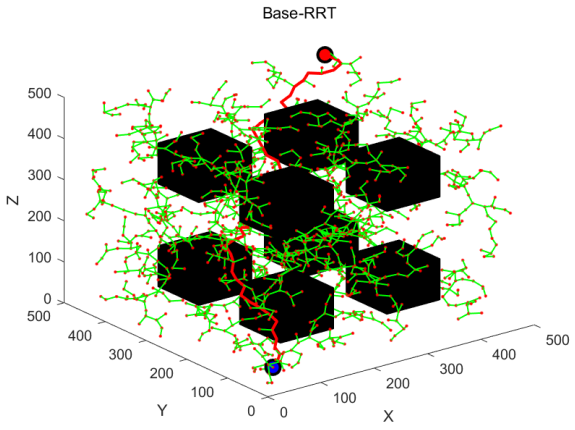


FIGURE 5. RRT search space expansion process. The blue point is the start. Red big-point is the goal. Red small-points is search tree nodes. Black geometry is an obstacle. The green lines are the growth path of the RRT tree search in space. The red line is the path that the RRT searches for the goal.

error interval; "Reached" means that the new node reaches the error interval of the goal node, that is, the path planning is completed; "Trapped" represents a collision during the expansion process, the expansion fails; "Graph" represents the generation of the search tree path map [23]

The searchability of RRT is powerful. However, the randomness and blindness caused by random sampling have obvious shortcomings. Figure 5 shows the search process of the RRT in 3D space. As space expands, the RRT tree needs to traverse the entire space. Although the algorithm can trace the path, a lot of computing resources and time are consumed.

B. HEURISTIC PROBABILITY STRATEGY

Heuristic probability is referred by heuristic strategy in the process of random sampling. In order to avoid searching for invalid areas, the RRT tree is extending to the goal within the probable range. If the collision is not detected in the direction, it will be stored in the RRT tree as a new node [24]–[26].

Although the method has clear directivity to the goal, it is easy to collide around obstacles or search failure when space has bigger obstacles. In order to avoid such a situation, a bias-goal factor is proposed to influence the distribution of random points in the path planning process.

C. BIAS-GOAL FACTOR STRATEGY

In basic RRT, the process of growing a new node q_{new} is configuring as

$$q_{new} = q_{near} + \delta \frac{\overrightarrow{q_{rand}} - \overrightarrow{q_{near}}}{|q_{rand} - q_{near}|} \tag{1}$$

The bias-goal factor φ is used to control the direction and distance of random points. The idea of the artificial potential field is combined in this strategy [27]. When the RRT tree is far away from the goal, the heuristic probability is used as guidance to avoid oscillating obstacles. When approaching the goal, the bias-goal factor can guide and quickly converge. So, adding bias-goal factor φ, the new node q_{new} is reconfigured as

$$q_{new} = q_{near} + \delta \frac{\overrightarrow{q_{rand}} - \overrightarrow{q_{near}}}{|q_{rand} - q_{near}|} + \varphi \frac{\overrightarrow{q_{goal}} - \overrightarrow{q_{near}}}{|q_{goal} - q_{near}|} \tag{2}$$

Let the angle between $\overrightarrow{q_{rand}q_{near}}$ and $\overrightarrow{q_{goal}q_{near}}$ is β. And the angle between $(\overrightarrow{q_{new}q_{near}} + \overrightarrow{q_{goal}q_{near}})$ and $\overrightarrow{q_{goal}q_{near}}$ is α. In the process of random sampling, it can be divided into the following three cases without obstacles as shown in Figure 6. When $\beta \in (0, \pi/2]$ q_{new} is closer to q_{goal} than q_{rand}. The search point can be more biased toward the goal, thereby improves search efficiency. When $\beta \in (\pi/2, \pi)$ q_{new} is closer to q_{goal} than q_{rand}. The moving step size will be reduced but the search invalid space can also be dropped. When $\beta = \pi$ which is the same as $\beta \in (\pi/2, \pi)$. It is necessary to control $\delta > \varphi$ to avoid the local minimum solution. In this way, the gravitational potential field will not lead to local minimum solutions in a complex environment. Meanwhile, it also improves the efficiency of the search process.

The bias-goal factor φ is configuring as

$$\varphi = \sum_{i=1}^{n} a_i e^{b_i x} \tag{3}$$

where x is $\left\|\overrightarrow{q_{goal}q_{near}}\right\|$.

PBG-RRT algorithm pseudo-code is shown in Algorithm 2. "Steer" is counted according to the heuristic probability strategy, and "MixSteer" is calculated according to the bias-goal factor strategy.

Algorithm 2 BG-RRT

Input:
Initial configuration q_{start} q_{goal} *obstacle*
Number of sampling point K
Heuristic probabilistic P
Step size δ
Bisa φ

Output:
RRT graph G
Vertices of tree *nodes*

Initialize all Parameters
$nodes = q_{start}$;
for $i = 1$ *to* K **do**
  q_{rand} = Sample();
  $q_{near} \leftarrow$ Nearest($nodes$, q_{rand});
  if $p > P$ **then**
    $q_{new} \leftarrow$ Steer(q_{near}, q_{goal}, δ);
  else
    $q_{new} \leftarrow$ MixSteer(q_{near}, q_{goal}, q_{rand}, φ, δ);
  if *ObstacleFree*(q_{near}, q_{new}) **then**
    $nodes$.add(q_{new});
    return Advanced;
if $dist(q_{new} - q_{goal}) < Error$ **then**
  return Reached;
else
  return Trapped
final ;
return *Graph*;

IV. ITRAJECTORY PLANNING

A. RAREFY TREATMENT

RRT tree nodes are increasing as space expands and path points of the trajectory plan are increasing accordingly. This is especially noticeable in 3D space. Moreover, the concavity and convexity of the curve are increased as the control points increase by B-spline fitting. So that the distance, time and manipulator energy consumption of a trajectory plan is also increasing accordingly. Therefore, it is important to minimize the control points for trajectory planning while maintaining the original curve characteristics.

In this paper, the Douglas-Peucker algorithm is adopted, and the algorithm steps are as follows. As shown in Figure 7, the AG is connected by a straight line between the first and last points of path planning. The point where other path points reached the maximum distance of line AG is C. And, the distance between point C and AG line is d. Given rarefy coefficient λ and d, if d is small, point C will be deleted. If d is large, point C will be reserved as a key point. Then C divides the original path into two segments AC and CG. Repeated the above steps. The resulting points are the path points after rarefying.

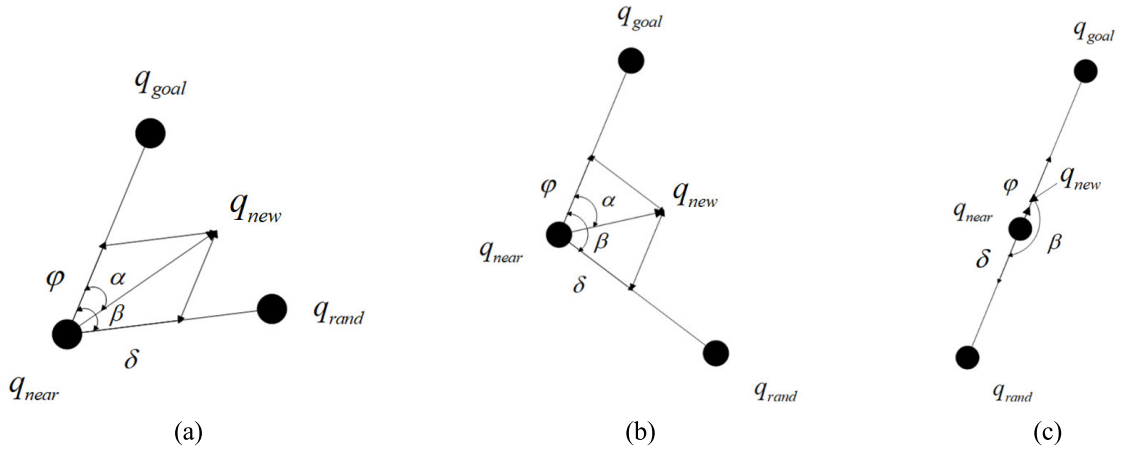


FIGURE 6. Three cases are combining with the bias-goal factor. q_{new} is the point of combining δ and φ. (a) Ability to approach the goal while increasing search efficiency under the influence of bias-goal factor. (b) Correct the direction of random point as much as possible toward the goal. (c) Need to control $\varphi > \delta$ to avoid falling into the local minimum.

B. NON-UNIFORM B-SPLINE FITTING

Due to the randomness and blindness of RRT, the continuous segments of planned path points are usually sawtooth. In this case, the manipulator moves unstable and oscillatory. So that the drive motor is impaired by a sudden change in acceleration [28].

Under the PBG-RRT algorithm, path points distribute dense at the beginning process, but end process sparse. In this paper, three-order Non-Uniform B-Spline interpolation is used to optimize trajectory. It applies to high-dimensional space that manipulator effectively solves joint smoothing problems. The foundation is laid for the manipulator to execute complex work tasks.

The B-spline curve equation as

$$p(u) = \sum_{i=o}^{n} d_i N_{i,k}(u) \tag{4}$$

where $d_i(i = 1, 2, \cdots, n)$ are control points. $N_{i,k}$ is the K-order normal B-spline basis function. The non-uniform node vector is parameterized by chord length to implement the parameterization process for control points. Figure 8 is a comparison of uniform nodes and non-uniform nodes.

The first node and the last node have k+1 repetition in Figure 8 (b). Different types of B-spline curves have different node vectors U.

According to the Riesenfeld method, let the sides of the polygon are in order: $l_i = |d_i - d_{i-1}|(i = 1, 2, \ldots, n)$. The total side length is $L = \sum_{i=1}^{n} l_i$. The even-order (5), as shown at the bottom of this page. The odd-order (6), as shown at the bottom of the next page. The basic function of B-spline is usually the Cox-deBoor recursion formula as

$$\begin{cases} N_{i,0}(u) = \begin{cases} 1, & u_i \le u \le u_{i+1} \\ 0, & others \end{cases} \\ N_{i,k} = \dfrac{u - u_i}{u_{i+k} - u_i} N_{i,k-1}(u) + \dfrac{u_{i+k+1} - u}{u_{i+k+1} - u_{i+1}} N_{i+1,k-1}(u) \\ \dfrac{0}{0} = 0; \\ 0 \times \infty = 0 \end{cases} \tag{7}$$

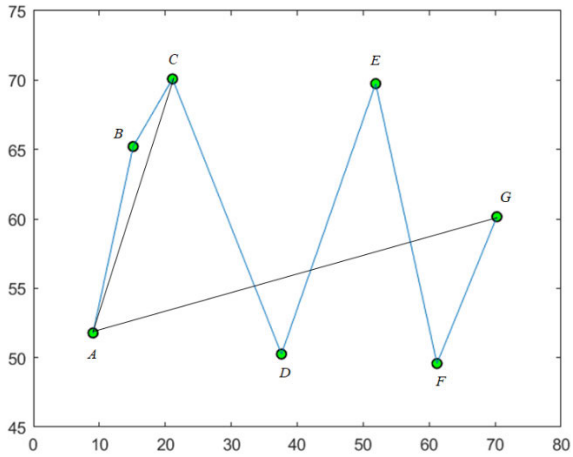


FIGURE 7. Schematic diagram of the Douglas-Peucker algorithm. Green points are path points. Point A is the start of the path. Point B is the end of the path.

$$U = \left[\underbrace{0,0,\ldots,0}_{k+1}, \frac{(\sum_{j=1}^{k/2} l_j) + \frac{l_{k/2+1}}{2}}{L}, \frac{(\sum_{j=1}^{k/2+1} l_j) + \frac{l_{k/2+2}}{2}}{L}, \ldots, \frac{(\sum_{j=1}^{n-k/2-1} l_j) + \frac{l_{n-k/2}}{2}}{L}, \underbrace{1,1,\ldots,1}_{k+1} \right] \tag{5}$$

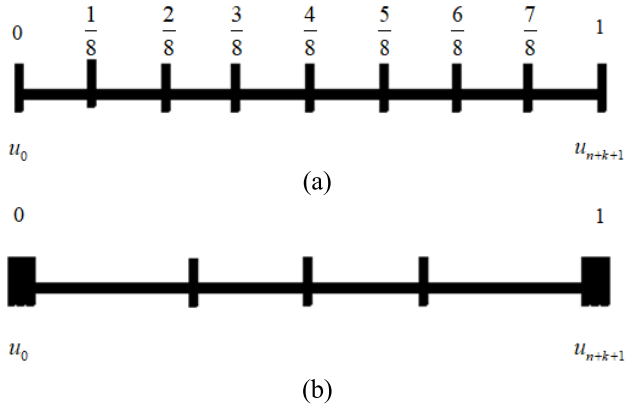


FIGURE 8. The different between uniform node and non-uniform node. (a) The uniform nodes are divided equally on the number axis of the unit. (b) Non-uniformity means that the range of influence vertices can be changed. Used to fit irregular curves with uneven distribution points.

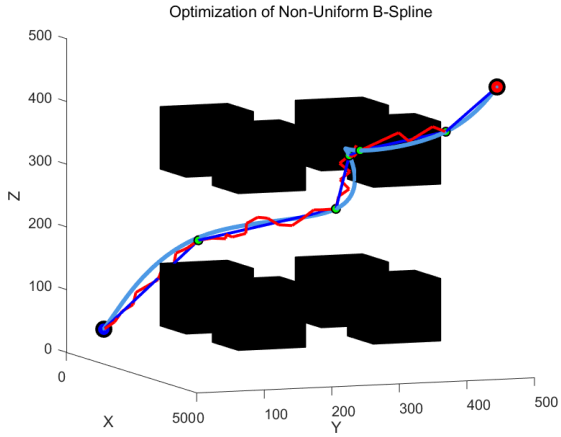


FIGURE 9. The track after optimization by non-uniform B-spline. The red line is a path planning trajectory. The blue line is path point line drawing after rarefying. The cyan line is the three-order Non-Uniform B-spline fitting trajectory. Green points are path points.

Path points are obtained during the path planning process. After rarefying, control points $d_i (i \in q_{curve})$ need to be treated in the reverse solution. The three-time B-spline of reverse solution equation is as

$$\begin{bmatrix} N_{1,3}(u_3) & N_{2,3}(u_3) & & N_{0,3}(u_3) \\ N_{1,3}(u_4) & N_{2,3}(u_4) & N_{3,3}(u_4) & \\ \ddots & \ddots & \ddots & \\ & N_{n-4,3}(u_n) & N_{n-3,3}(u_n) & N_{n-2,3}(u_n) \\ N_{\text{n-1},3}(u_{n+1}) & & N_{n-3,3}(u_{n+1}) & N_{n-2,3}(u_{n+1}) \end{bmatrix} \times \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-3} \\ d_{n-2} \end{bmatrix} = \begin{bmatrix} q_0 \\ q_1 \\ \vdots \\ q_{n-4} \\ q_{n-3} \end{bmatrix} \tag{8}$$

where q_i are path points.

As for the elements are values of the basic function in the coefficient matrix, they are only related to the node vectors U. Equation (8) is simplified as

$$\begin{bmatrix} b_1 & c_1 & & a_1 \\ a_2 & b_2 & c_2 & \\ \ddots & \ddots & \ddots & \\ & a_{n-3} & b_{n-3} & c_{n-3} \\ c_{n-2} & & a_{n-2} & b_{n-2} \end{bmatrix} \tag{9}$$

Each parameter as

$$\begin{cases} a_i = \dfrac{(\Delta_{i+2})^2}{\Delta_i + \Delta_{i+1} + \Delta_{i+2}} \\ b_i = \dfrac{\Delta_{i+2}\,(\Delta_i + \Delta_{i+1})}{\Delta_i + \Delta_{i+1} + \Delta_{i+2}} + \dfrac{\Delta_{i+1}\,(\Delta_{i+2} + \Delta_{i+3})}{\Delta_{i+1} + \Delta_{i+2} + \Delta_{i+3}} \\ c_i = \dfrac{(\Delta_{i+1})^2}{\Delta_{i+1} + \Delta_{i+2} + \Delta_{i+3}} \\ e_i = (\Delta_{i+1} + \Delta_{i+2})\, q_{i-1}, \end{cases} \quad i = 1, 2, \ldots, n-2$$

Then a smooth trajectory is generated by control points. As shown in Figure 9, trajectory optimization is realized on the original path planning.

After fitting by Non-Uniform B-spline, the question which obtained trajectory from fitting collides with the obstacle may exist. This case which is defined as planning failure happens in a pretty complex environment. As far as a large number of simulations in this paper, the success rate of planning is nearly closing to 100% .

V. SIMULATION AND EXPERIMENT

In this section, in order to verify the validity and reliability of PBG-RRT, set two 500 × 500 × 500 maps. The validity is verified in Map-1, and the reliability is proved in Map-2. All experiments are performed on an Intel Core i5-8265U 1.8GHz computer with 8GB of memory. Set start point as (40, 40, 40) and goal point as (460, 460, 460). Set heuristic probability $P = 0.1$.

$$U = \left[\underbrace{0,0,\ldots,0}_{k+1}, \frac{\sum_{j=1}^{(k+1)/2} l_j}{L}, \frac{\sum_{j=1}^{(k+1)/2+1} l_j}{L}, \ldots, \frac{\sum_{j=1}^{n-(k+1)/2} l_j}{L}, \underbrace{1,1,\ldots,1}_{k+1} \right] \tag{6}$$

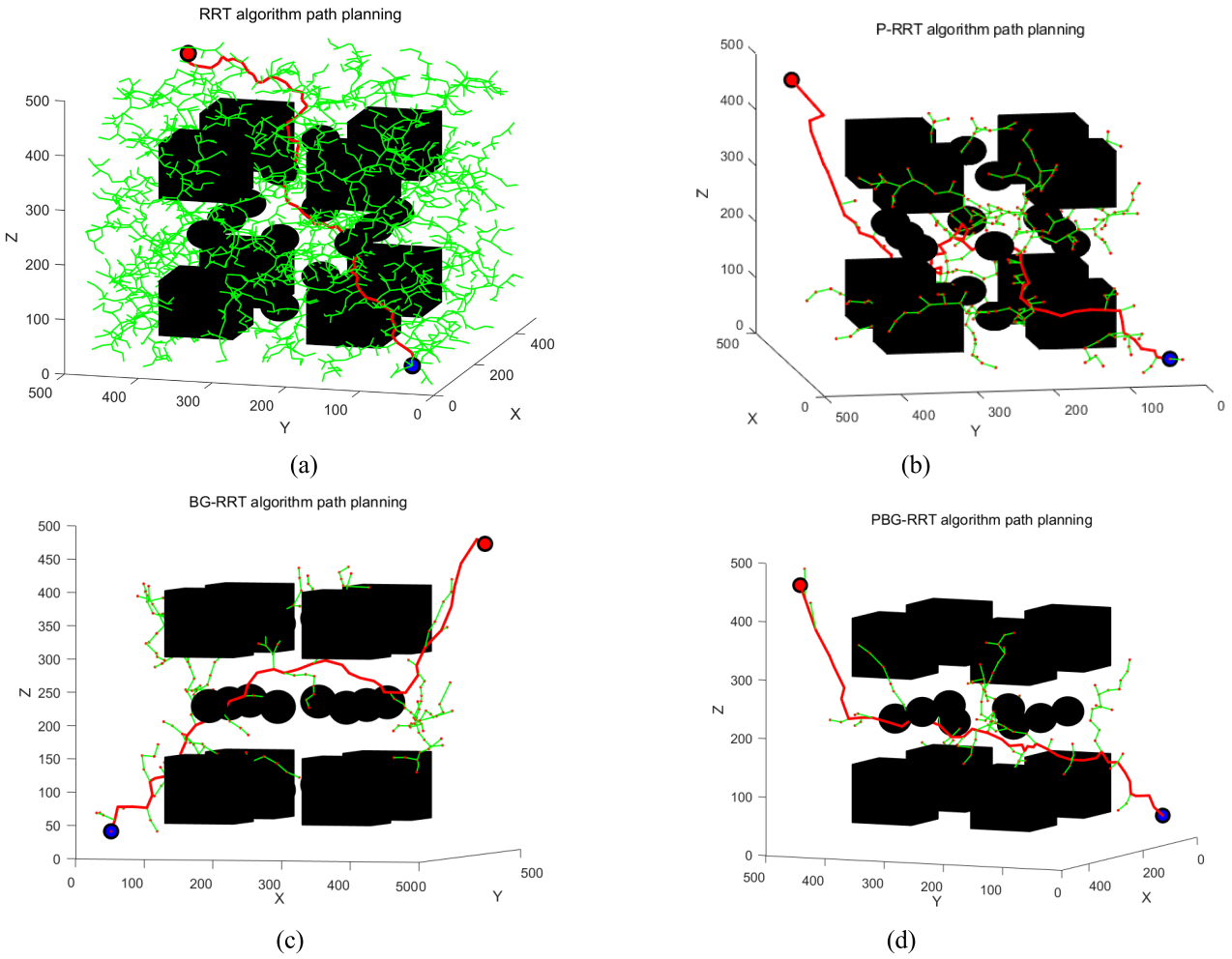


FIGURE 10. Compare with four algorithms of path planning. Green point is the start point. Red big-point is the goal. Red small-points is search tree nodes. The green path is the echo algorithm search path. The red path is the path planning trajectory. (a)–(d) show the PBG-RRT searches paths with different algorithms.

After verifying by Matlab, the whole motion planning Strategy is simulated in ROS. Specifically, adjust experiment parameters in the "src" files and set up the simulation environment in Rviz. Finally, the feasibility of motion planning is demonstrated by the above simulations.

A. ALGORITHM VALIDATION IN MATLAB

RRT search time increases with the complexity of space. In order to verify the validity of PBG-RRT, other algorithms are compared in Map-2. Such as RRT, P-RRT and BG-RRT (BG-RRT is PBG-RRT without heuristic probability). Set step size $\delta = 20$ and simulation 1000 times. The average data are shown in Table 1.

Table 1 indicates, RRT average search time is the least, and then the P-RRT also shows good searchability. Though, RRT is slow to search in complex space. Other algorithms compared with PBG-RRT, the average length of path planning is reduced by 11%-27%. Also, the average number of nodes in the path is dropped by 16% -39%. Especially, the search efficiency of PBG-RRT is increased by 217% compared with P-RRT.

TABLE 1. Four algorithms experimental data.

Algorithm	Avg. Number of nodes in the tree	Avg. Number of nodes in the path	Avg. Path length	Avg. Time of the path planning (s)
RRT	1004.107	67.376	1323.539	12.515
P-RRT	409.018	62.426	1233.877	0.201
BG-RRT	263.752	49.367	1087.129	0.684
PBG-RRT	129.092	41.019	971.332	0.075

Because the bias-goal factor converges quickly, PBG-RRT has a significant improvement in searching valid space. Comparing with P-RRT, the average search time of PBG-RRT increased by 168%. Likewise, comparing with BG-RRT, the average search time of PBG-RRT is greatly increased. That means the bias-goal factor performs poor at the beginning of processing, but it is continuously enhanced in the process of approaching. In 1000 simulation experiments, the variance of BG-RRT search time is large. Result from in the early stage, the bias-goal factor still has certain blindness.

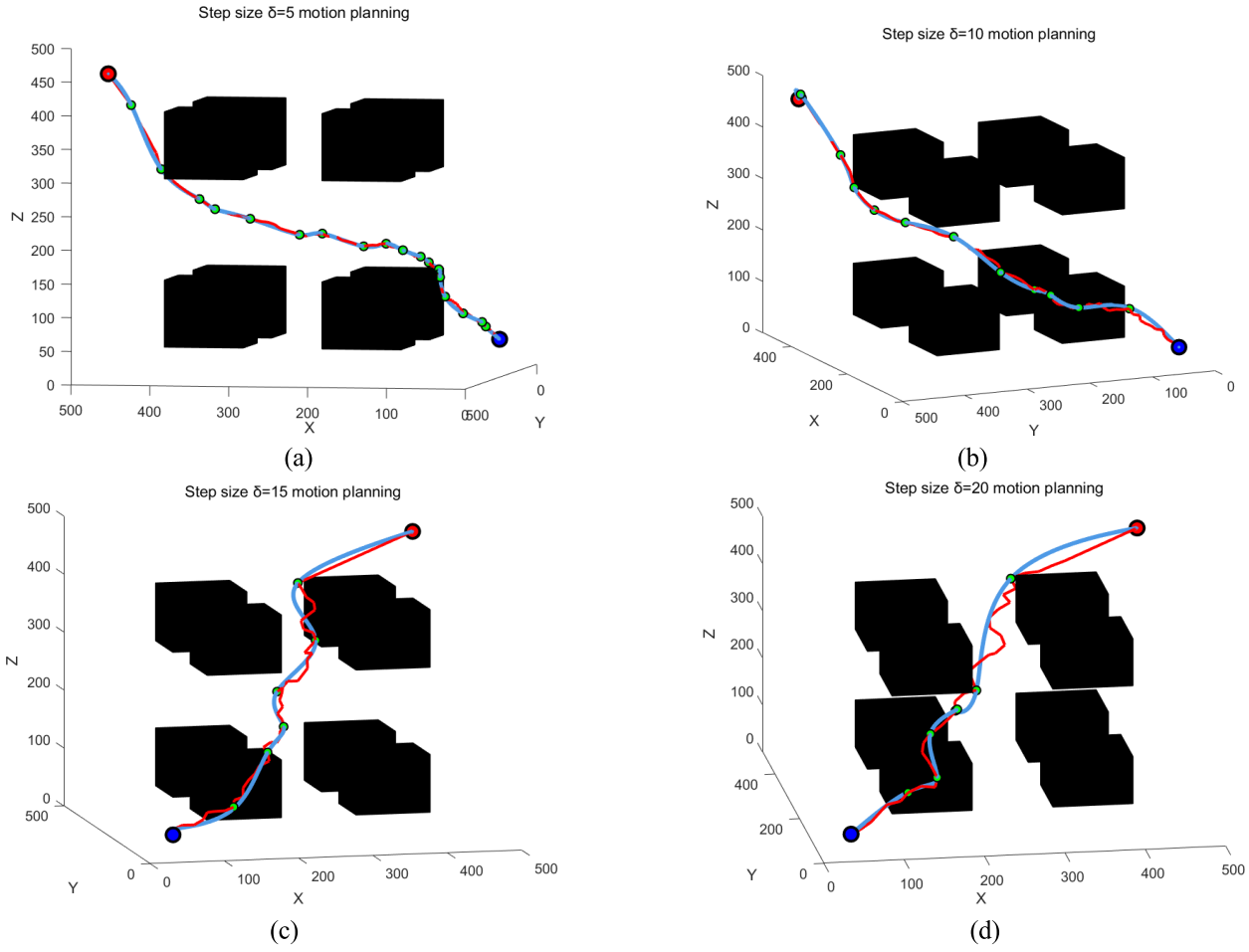


FIGURE 11. **Compare with four steps of motion planning. The red line is the path planning trajectory. The cyan line is the three-order Non-Uniform B-spline fitting trajectory. Green points are path points. (a)–(d) show path points are decreasing as δ increases. And the difference between curves is also increasing.**

According to Figure 10, other algorithms oscillate near obstacles in the complex environment compared with PBG-RRT. Searching does not have directivity, even lots of sawtooth paths are planned and invalid spaces are searched. Obviously, the number of nodes in the search process and calculations in invalid space are dropped by PBG-RRT. During the search process, PBG-RRT is pointing to the goal constantly and avoiding shocking around obstacles.

Through the above comparative analysis, PBG-RRT combines the advantages of bias-goal factor and heuristic probability and make up for each disadvantage. Enable to converge instantly and avoid falling into the local minimum. So, PBG-RRT has absolute advantages in complex environments. Compared with other algorithms, PBG-RRT meets validity in a complex environment.

B. ALGORITHM RELIABILITY IN MATLAB

The reliability of PBG-RRT is proved by different step sizes. Take 1000 times simulation with different δ in Map-1. The results are shown in Table 2. Similar trajectories under different δ are selected and shown in Figure 13.

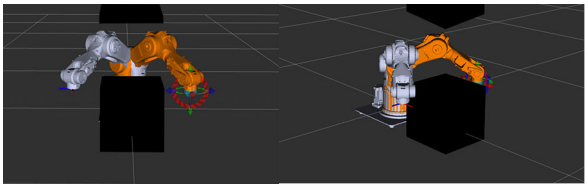
On the basis of Table 2. When step size is longer, the searchability is stronger, the average of nodes in a path is fewer and the average path length is shorter. For rarefying, path points are dropped by 78%-83% while ensuring the original curve characteristics. Moreover, the average distance after the trajectory planning is reduced by 7% -10% compared with path planning.

When $\delta = 5$, the consumption of average time has increased sharply, which is caused by the bias-goal factor. In other words, the bias-goal factor has little effect when moving away from the goal. In this time, the heuristic probability mainly affects to proximity goal. When approaching the goal, it is quickly converging by bias-goal factor. Hence, suitable step size is important for the reliability of PBG-RRT.

In Figure 11, the path point decreases along with δ increases. But the deviation is brought on curves. At the same time, as the step size increases, the length of the interpolated trajectory is continuously reducing. Because of the trajectory interpolated by Non-Uniform B-spline after rarefying.

TABLE 2. Four kinds of δ with PBG-RRT experimental data.

Step size	Avg. Number of nodes in the tree	Avg. Number of nodes in the path	Avg. Number of path points after rarefying	Avg. Path planning length	Avg. Path length after spline fitting	Avg. Time of the motion planning(s)
5	782.386	104.794	19.697	1008.059	930.995	1.259
10	273.428	64.422	14.820	971.542	921.540	0.303
15	202.397	53.387	9.682	978.772	889.139	0.154
20	129.092	41.019	8.757	969.332	898.577	0.115



(a) (b)

FIGURE 12. The Robot-Anno manipulator in Rviz. It shows the start and goal of the manipulator pose. The white one is the manipulator start pose. The yellow one is the manipulator goal pose. Black geometry is an obstacle, which needs to avoid.

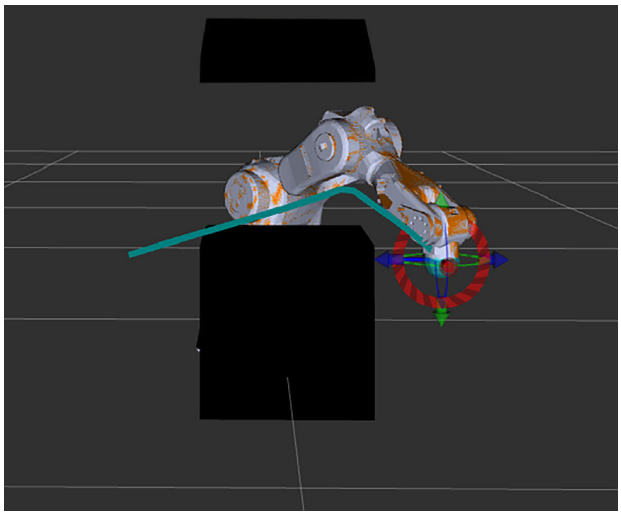


FIGURE 13. Manipulator motion planning process. Start state of manipulator coincides with the goal state. This indicates that motion planning is completed. The green line, which is smooth and less inflection point, is the trajectory of motion planning.

The above methods can effectively avoid margin planning caused by large-scale path planning. Meanwhile, the reliability of PBG-RRT is proofed.

C. ROS EXPERIMENT

Robot-Anno manipulator is used as the research object in this paper. In this case, the visualization tool Rviz for simulation is performed in ROS, which simulated with a complete motion planning solution. Specifically, a manipulator simulation environment is constructed in Rviz. Then, set obstacles and poses as shown in Figure 12. Afterward, set the start and goal pose of manipulator end-effector parameters

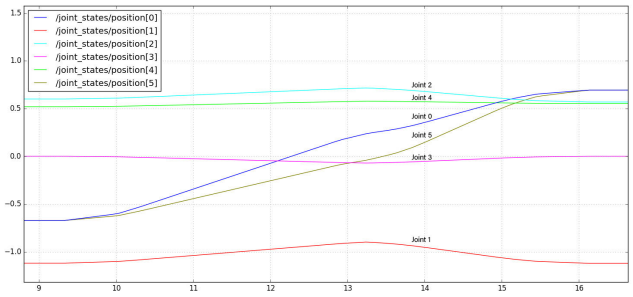


FIGURE 14. Changes in each joint of motion planning. The lines of six colors in the figure represent the changes in the motion planning of each joint. Each of the joint changes stable without sudden change.

TABLE 3. Each joint angle under the start pose and the goal pose.

	Joint 0	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5
start pose	-0.67	-1.12	0.60	0.00	0.52	-0.67
goal pose	0.69	-1.12	0.57	0.00	0.55	0.69

TABLE 4. The average time of the different algorithms under the same environment.

	P-RRT	TRRT	RRT-Connect	RRT-Star	BIT-RRT	PBG-RRT
Avg. Time of planning(s)	0.637	3.352	0.521	5.028	0.025	0.019

as (units: m)

$$P_{start} = \begin{bmatrix} 0.0002 & -1.0000 & -0.0001 & 0.3113 \\ 1.0000 & 0.0002 & 0.0001 & -0.2502 \\ -0.0001 & -0.0001 & 1.0000 & 0.2143 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (10)$$

$$P_{goal} = \begin{bmatrix} 0.0002 & -1.0000 & -0.0001 & 0.3056 \\ 1.0000 & 0.0002 & 0.0001 & 0.2514 \\ -0.0001 & -0.0001 & 1.0000 & 0.2073 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (11)$$

The angle of each joint as shown in Table 3 (units: radian).

Using the PBG-RRT algorithm combined with three-order Non-Uniform B-spline processing after rarefying. A smooth path is gotten in simulation as shown in Figure 13. The manipulator trajectory is smooth during the obstacle avoidance process. Simultaneously, each angle of joint changes smoothly during the simulation without any sudden change

or shock in Figure 14. Also, the time of different algorithms is simulated with twenty times in the same environment as shown in Table 4.

VI. CONCLUSION

Several improvements have been proposed for shortcomings of RRT and manipulator motion planning optimization. Firstly, bias-goal factor and heuristic probability strategy are combined for the PBG-RRT algorithm. The speed of planning is greatly improved in 3D space with obstacles. At the same time, the phenomenon of the local minimum is averted. Then the path points are obtained by rarefying. Redundant path points are minimized while maintaining the original curve characteristics. The phenomenon of oscillating near obstacles during path planning is avoided. Meanwhile, the number of inflection points can be reduced generated from Non-Uniform B-spline fitting. According to the density distribution of the path point, a three-order Non-Uniform B-spline is used. Hence, the smooth end-effector trajectory is obtained. The whole set of motion planning strategy is basing on the complementary of algorithms. This is rarely doing by previous scholars, but it is important for a manipulator. Finally, the simulation is performed on ROS. The motion planning strategy is reliable from the time of planning and the changes in each angle of the joint.

There are still many works worth studying about this motion planning strategy. The obstacle expansion still occupies a part of the effective space for path planning search. In addition, combined with dynamic obstacle avoidance, it can better meet more scenarios.

REFERENCES

[1] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey," *Robot. Auto. Syst.*, vol. 86, pp. 13–28, Dec. 2016.

[2] P. K. Mohanty, K. Prases, and D. R. Parhi, "Controlling the motion of an autonomous mobile robot using various techniques: A review," *J. Adv. Mech. Eng.*, vol. 1, no. 1, pp. 24–39, 2013.

[3] P. Raja and S. Pugazhenthi, "Optimal path planning of mobile robots: A review," *Int. J. Phys. Sci.*, vol. 7, no. 9, pp. 1314–1320, Feb. 2012.

[4] A. Gasparetto and V. Zanotto, "A new method for smooth trajectory planning of robot manipulators," *Mechanism Mach. Theory*, vol. 42, no. 4, pp. 455–471, 2007.

[5] Q. C. Chen, S. Q. Zhu, and X. Q. Zhang, "Improved inverse kinematics algorithm using screw theory for a six-DOF robot manipulator," *Int. J. Adv. Robot. Syst.*, vol. 12, no. 10, p. 140, 2015.

[6] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, 2014.

[7] K. Wei and B. Ren, "A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved RRT algorithm," *Sensors*, vol. 18, no. 2, p. 571, 2018.

[8] R. C. Luo and C.-W. Kuo, "Intelligent seven-DoF robot with dynamic obstacle avoidance and 3-D object recognition for industrial cyber-physical systems in manufacturing automation," *Proc. IEEE*, vol. 104, no. 5, pp. 1102–1113, 2016.

[9] Z. C. He, Y. L. He, and B. Zeng, "Obstacle avoidance path planning for robot arm based on mixed algorithm of artificial potential field method and RRT," *Ind. Eng. J*, vol. 20, pp. 56–63, 2017.

[10] S. M. LaValle and J. J. Kuffner, Jr., "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, 2001.

[11] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. Int. Conf. Robotic Auto. (ICRA)*, San Francisco, CA, USA, Apr. 2000, pp. 995–1001.

[12] B. Boyacioglu and S. Ertugrul, "Time-optimal smoothing of RRT-given path for manipulators," in *Proc. ICINCO*, vol. 2, 2016, pp. 406–411.

[13] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[14] X. Cao, X. Zou, C. Jia, M. Chen, and Z. Zeng, "RRT-based path planning for an intelligent litchi-picking manipulator," *Comput. Electron. Agricult.*, vol. 156, pp. 105–118, Jan. 2019.

[15] H. J. Zhang, "Path planning of industrial robot based on improved RRT algorithm in complex environments," *IEEE Access*, vol. 6, pp. 53296–53306, 2018.

[16] X. Y. Wang, "Bidirectional potential guided RRT* for motion planning," *IEEE Access*, vol. 7, pp. 95034–95045, 2019.

[17] 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. 2014.

[18] J. Z. Song and B. Dai, "An improved RRT path planning algorithm," *Acta Electron. Sinica*, vol. 38, no. 2A, pp. 225–228, 2010.

[19] K. Solovey, O. Salzman, and D. Halperin, "Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning," in *Algorithmic Foundations of Robotics XI*. Cham, Switzerland: Springer, 2015, pp. 591–607.

[20] I. Aguinaga, D. Borro, and L. Matey, "Parallel RRT-based path planning for selective disassembly planning," *Int. J. Adv. Manuf. Technol.*, vol. 36, nos. 11–12, pp. 1221–1233, 2008.

[21] X. L. Zhang, "Dynamic path planning algorithm for a mobile robot based on visible space and an improved genetic algorithm," *Int. J. Adv. Robot. Syst.*, vol. 13, no. 3, p. 91, 2016.

[22] M. Wasielica and D. Belter, "RRT-based motion planner and balance controller for a biped robot," in *Advances in Cooperative Robotics*. Singapore: World Scientific, 2017, pp. 404–411.

[23] A. Tahirovic and G. Magnani, "A roughness-based RRT for mobile robot navigation planning," *IFAC Proc. Volumes*, vol. 44, no. 1, pp. 5944–5949, 2011.

[24] J. G. Cao, "Robot global path planning based on an improved ant colony algorithm," *J. Comput. Commun.*, vol. 4, no. 2, p. 11, 2016.

[25] M. Ayyıldız and K. Çetinkaya, "Comparison of four different heuristic optimization algorithms for the inverse kinematics solution of a real 4-DOF serial robot manipulator," *Neural Comput. Appl.*, vol. 27, no. 4, pp. 825–836, 2016.

[26] P. D. H. Nguyen, U. Pattacini, G. Metta, and M. Hoffmann, "A fast heuristic Cartesian space motion planning algorithm for many-DoF robotic manipulators in dynamic environments," in *Proc. IEEE/RAS Int. Conf. Hum. Robot.*, Cancún, Mexico, Nov. 2016, pp. 884–891.

[27] Y. Han and G. D. Liu, "Mobile robot motion planning based on potential field in dynamic environment," *Robot*, vol. 28, no. 1, pp. 45–49, 2006.

[28] S. Kucuk, "Optimal trajectory generation algorithm for serial and parallel manipulators," *Robot. Comput.-Integr. Manuf.*, vol. 48, pp. 219–232, Dec. 2017.

CHENGREN YUAN is currently pursuing the master's degree in mechanical engineering with the College of Power Engineering, Naval University of Engineering, Wuhan, China. His research interests include underwater robot and motion planning.

WENQUN ZHANG received the Ph.D. degree from the College of Weapons Engineering, Naval University of Engineering, Wuhan, China, in 2004. He is currently an Associate Professor with the Naval University of Engineering. His research interest includes the optimized design of ship machinery.

GUIFENG LIU received the Ph.D. degree from the College of Power Engineering, Naval University of Engineering, Wuhan, China, in 2013. He is currently an Associate Professor with the Naval University of Engineering. His research interests include ship machinery maintenance and equipment support technology.

XINGLONG PAN received the Ph.D. degree from the College of Power Engineering, Naval University of Engineering, Wuhan, China, in 2013. He is currently a Lecturer with the Naval University of Engineering. His research interests include mechanical and electrical equipment maintenance support, and health management and robot motion planning.

XIAOHU LIU received the Ph.D. degree from the College of Electrical Engineering, Naval University of Engineering, Wuhan, China, in 2005. He is currently an Associate Professor with the Naval University of Engineering. His research interests include embedded systems and intelligent detection technology.