Robotica (2022), **40**, pp. 1855–1867
doi:10.1017/S0263574721001417

CAMBRIDGE
UNIVERSITY PRESS

RESEARCH ARTICLE

Effective motion planning of manipulator based on SDPS-RRTConnect

Junxiang Xu and Jiwu Wang*

School of Mechanical and Electronic Engineering, Beijing Jiaotong University, Beijing, Haidian District, China
***Corresponding author.** E-mails: 19121251@bjtu.edu.cn, jwwang@bjtu.edu.cn

Received: 25 April 2021; **Revised:** 16 July 2021; **Accepted:** 27 August 2021; **First published online:** 4 October 2021

Keywords: motion planning, RRT, RRTConnect, collision detection, sparse dead points saved

Abstract
In order to improve the speed of motion planning, this paper proposes an improved RRTConnect algorithm (SDPS-RRTConnect) based on sparse dead point saved strategy. Combining sparse expansion strategy and dead point saved strategy, the algorithm can reduce the number of collision detection, fast convergence, avoid falling into local minimum, and ensure the completeness of search space. The algorithm is simulated in different environments. The results show that in complex environments, the sparse dead point saved strategy plays a good effect. In simple environments, the greedy connection strategy works well. Compared with the standard RRT algorithm, the standard RRTConnect algorithm, and the SDPS-RRT algorithm, the SDPS-RRTConnect algorithm has the shortest planning time, and it is suitable for both simple and complex environments. The 500 experiments show that the algorithm has strong robustness. The actual robot experiments show that the path planned by SDPS-RRTConnect algorithm is effective.

1. Introduction
With the rapid development of science and technology, robots have been widely used in various industries such as manufacturing and service industries [1–3]. However, due to the complexity of tasks, the traditional teaching methods are no longer suitable, and the autonomous motion planning of robots has become an urgent problem to be solved. The existing motion planning methods mainly fall into the following three types: The first type is graph search algorithm (A*, Dijkstra), which needs to discretize the configuration space to ensure search completeness and optimality, which limits the flexibility of the algorithm to A certain extent [4–5]. The second type is the artificial potential field method, which requires little computation, but is easy to fall into local minimum value when searching, and is difficult to adapt to the map of complex environment [6]. The other is the method based on random sampling, which can better solve the path planning problem in high-dimensional space and complex constraints, but it has higher requirements on the number and uniform distribution of random points, and has lower planning efficiency in confined space [7]. Rapid Exploration Random Tree (RRT) algorithm can be used to solve the high-dimensional space motion planning problems with complex constraints. It has the advantages of probabilistic integrity, easy implementation, and strong scalability, but it has problems such as large amount of computation and slow convergence rate in complex environment [8].

To solve the above problems, many scholars have improved the RRT algorithm. The improvement is mainly divided into two aspects. One aspect is to reduce the path length, the other aspect is to reduce the planning time. For the first aspect, Karaman and Frazzoli proposed the RRT* algorithm, which is an optimal planning method based on RRT [9]. It guarantees the asymptotic optimality of the algorithm by adding "nearest vertex" and "reconnect" in Basic-RRT. However, the planning process of the RRT* algorithm is time consuming, so it is impractical to plan for tasks with a certain time requirement. Subsequently, many methods were proposed to accelerate the convergence of RRT*, but the calculation time still could not meet the requirements of practical application [10–12]. On the other hand, Kuffner and Lavalle proposed the BI-RRT algorithm, which grows two trees from the initial state and the target state, respectively, thus improving the exploration speed and convergence speed of the algorithm [13]. On the basis of BI-RRT algorithm, they proposed RRTConnect to improve the efficiency of node expansion [14]. RRTConnect algorithm builds and extends the random tree from the starting point and the target point simultaneously, which greatly improves the convergence speed.

This paper focuses on the second purpose of the algorithm improvement, which is to reduce the computation time of the algorithm under the premise that the algorithm can be applied to complex environment. Common approaches include reducing the number of nodes in the tree and the number of collision detection. RRT-Biased improves the performance of the algorithm through target bias sampling with a certain probability (typically 1–10%) [15]. However, although the RRT-Biased algorithm appropriately reduces the number of nodes in the random tree, the computation cost is still large and is prone to falling into local minima. Chengren et al. based on the RRT-Biased algorithm and combining heuristic probability and biased target factors, proposed an improved PGB-RRT algorithm based on heuristic probabilistic biased target [16]. Although this algorithm can accelerate convergence and avoid falling into the local minimum, this algorithm is not completely avoided. It will still sample in the vicinity, and it is possible to avoid the local minimum after several attempts. In addition, the search effect of the algorithm in complex cases is not good, and the nodes are still too dense. Wang and X proposed a general node control autonomous path planning algorithm. By gradually changing the sampling area to guide exploration and improve search speed, a node control mechanism was introduced to reduce the expansion of invalid nodes and extract boundary nodes. However, this algorithm needs to select the value of node control factor, which has certain limitations [17]. Wei proposed an autonomous dynamic obstacle avoidance algorithm (S-RRT), which significantly improved the sampling speed and efficiency of RRT for directional node expansion [18]. Xinyu et al. integrated the two-way artificial potential field into the rapid detection of random tree stars, which improved the search speed of path planning, but the artificial potential field method was easier to enter the local minimum [19]. Zhang introduced regression mechanism to prevent excessive searching of configuration space, reduce The times of collision detection and improve planning efficiency [20]. Gitae Kang et al proposed a manipulator motion planning method based on target sampling. This method improves the search speed by limiting the distribution of sampling points, but the invalid detection area is large, and the nodes of this algorithm are relatively dense [21]. In addition, on the basis of BI-RRT and RRTConnect algorithms, Qiao proposed a sample-based multi-tree fusion algorithm for boundary detection, which overcomes the shortcomings of slow growth in complex environments by changing the growth and storage rules of random trees and achieves good results in complex environments [22].

RRT algorithm and RRTConnect algorithm need a lot of repeated and excessive search in a complex environment, resulting in the phenomenon of nodes being too close and edges crossing, as shown in Fig. 1. Excessive search will lead to a sharp increase in the number of collision detection, and collision detection will take up a lot of time for manipulator motion planning. Therefore, this paper proposes an improved RRTConnect algorithm (SDPS-RRTConnect) based on sparse dead point saved strategy. Combining sparse expansion strategy and dead point saved strategy, the algorithm can reduce the number of collision detection, fast convergence, avoid falling into local minimum and ensure the completeness of search space. By controlling the distance between nodes, the sparsity of the random tree is guaranteed, the repeated search is avoided and the search space is ensured. In addition, the dead point saved strategy is introduced to constrain the extension nodes of the tree, thus reducing the expansion of invalid nodes and greatly reducing the times of collision detection. The local minimum condition is avoided by not extending the node after it becomes dead. To make it easier to understand, we first describe the algorithm in 2D space, and then extend it to 3D space.

The rest of the paper is organized as follows. Section 2 overview the proposed method. On this basis, it is represented by flowchart. Section 3 introduces the improved RRT algorithm proposed in this paper in two parts. The first part introduces the sparse dead point saved strategy in two-dimensional (2D) environment and the second part gives the sparse dead point saved strategy in three-dimensional (3D)

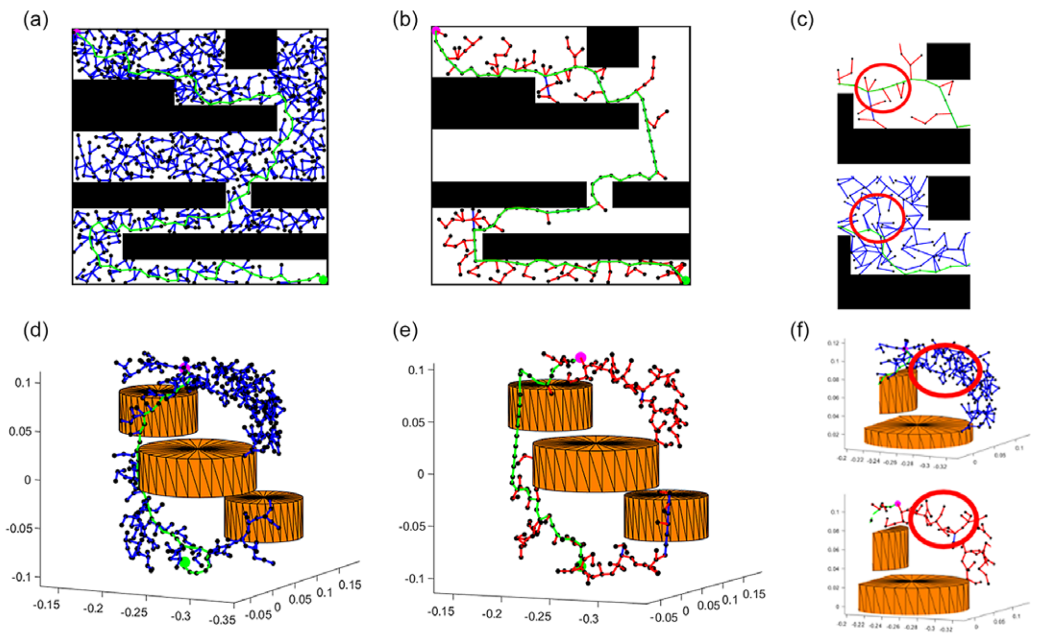


Figure 1. *Complex environment planning results.*

environment. In Section 4, the algorithm is simulated in 2D- and 3D space, and the simulation process and results are explained. In the fifth section, it is applied to the simulation of 6-DOF manipulator and verified by real robot experiments. Finally, Section 6 gives the conclusion and future work of this paper.

2. Methodology

The improved algorithm obtains a new node after sampling and expansion, and then looks for the node nearest to the new node in the tree. If the distance between the new node and the nearest node is less than the expansion step, the expansion fails. When the distance between the new node and the nearest node is greater than the extension step, the node closest to the new node in the saved dead point is looked for. If the distance between the new node and the nearest dead point is less than the extension step, the extension fails. When the distance of the new node to the nearest dead point is greater than the extension step, if the collision detection times of its parent node are greater than the extension times of its parent node, the extension fails. Otherwise, collision detection is carried out, and the node is added into the tree if the collision detection is successful, and the node is saved if the collision detection fails. SDPS in the algorithm is the threshold value of the node expansion times. Delta is the number of collision detection (including the number of successes and failures) for each node in the tree. The flowchart of this algorithm is shown in Fig. 2.

3. Sparse Dead Point Saved Strategy

In order to reduce the number of collision detection, accelerate the search speed of the RRT algorithm, and avoid repeated excessive search and falling into the minimum value, we proposed a sparse dead point saved strategy. In order to facilitate the description, we first explain the sparse dead point saved strategy in 2D environment, and then extend it to three-dimensional environment.

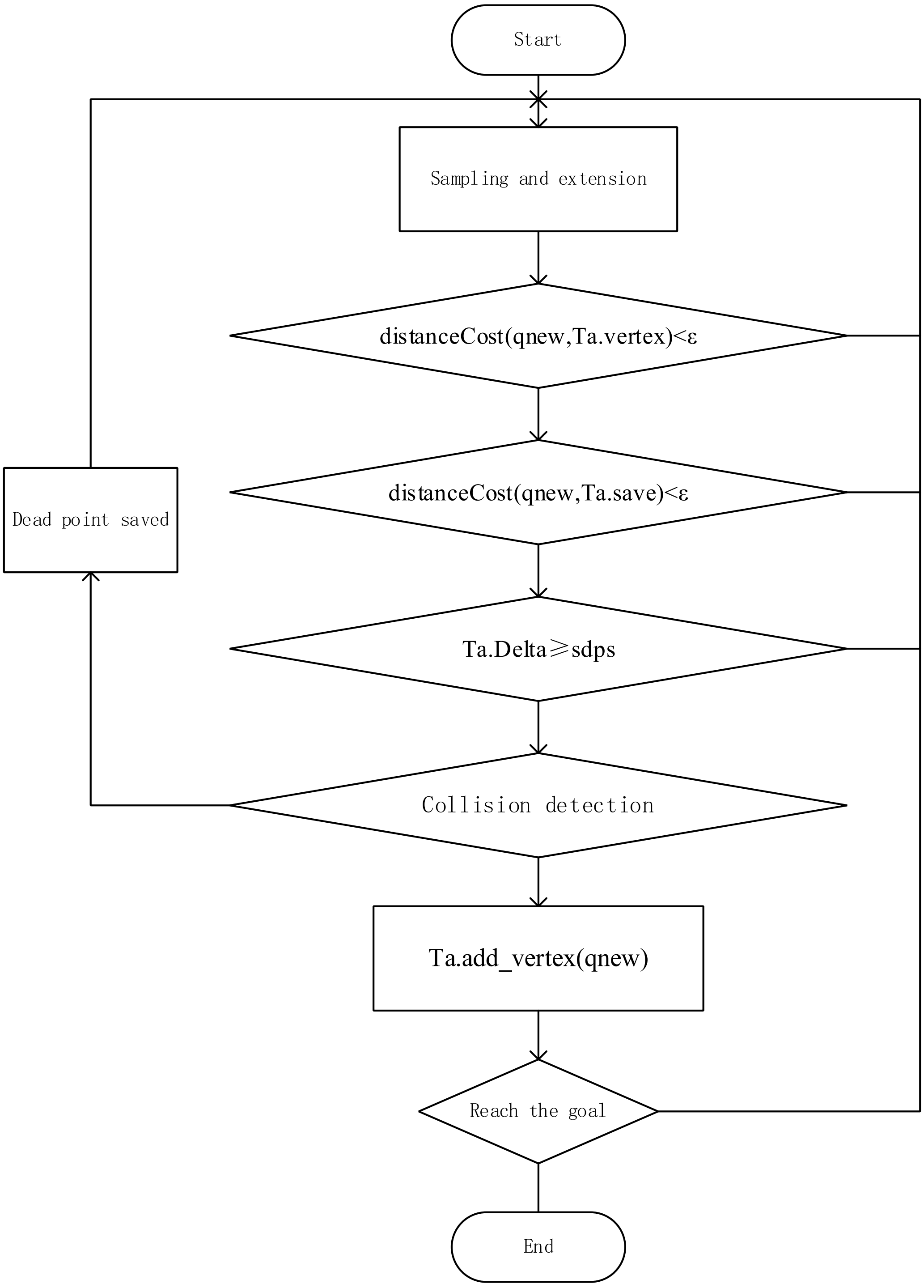


Figure 2. *Flowchart of algorithm.*

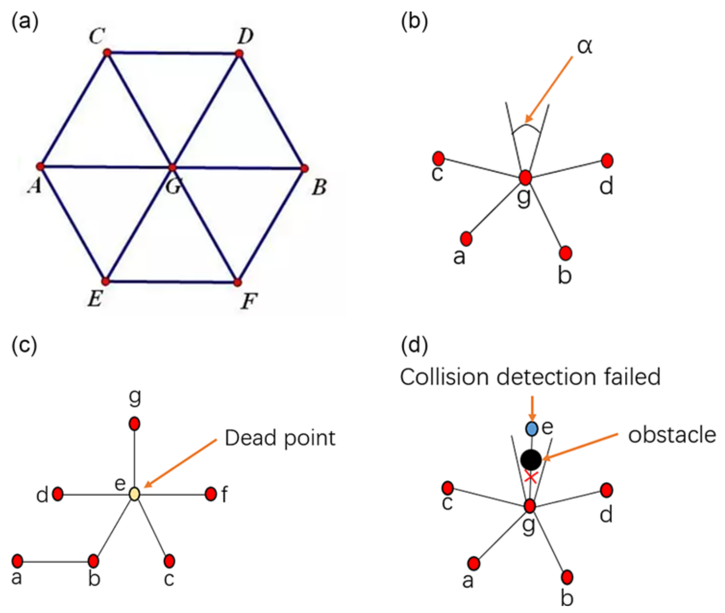


Figure 3. *Sparse dead points saved. (a) Node extension range, (b) repeat angle, (c) dead point, (d) collision detection failed child node.*

3.1. Two-dimensional sparse dead point saved strategy

As shown in Fig. 3(a), for a parent node G, it can extend 6 child nodes outwardly, and the distance between each child node should be greater than or equal to the extension step size, so as to ensure the sparsity of the extension tree. However, when a parent node extends 4 children nodes and the distance of each node is greater than the expansion step, the parent node will only be able to extend one child node. This child node will have an extended range of fan angles, which we call the repeat angle (α), as shown in Fig. 3(b). If random sampling points within the range of the repeat angle and there are obstacles within the range of the repeat angle, the number of collision detection will be greatly increased, thus increasing the expansion time. In this case, we propose a dead point saved strategy. For the child node that fails collision detection, we save itself and this extension to the extended number of its parent node, so as to ensure that the next sampling node will not appear in the vicinity again. For the node with successful collision detection, we add it into the tree and save the expanded times of its parent node. If a node has been collision detected more than 6 times, it is a dead point and will not be extended. In addition, the yellow node e in Fig. 3(c) is a dead point. Although it has only five edges, since the angle between nodes d, g and f is greater than 60 degrees, the randomly sampled nodes will inevitably not meet the above conditions, so node e is a dead point. But for the sake of uniformity, we still specify that each node can only be extended six times. In general, a node is extended by a parent node, that is, a node will first have a parent node, then the node can be extended no more than 5 times. So, for the starting point, the number of extensions is 6. For the rest of the nodes, the number of extensions is 5. Nodes whose collision detection times exceed their respective expansion times will no longer have expansion opportunities and become dead points. It should be noted that the 'save' here refers to both the save of the dead point and the save of the child nodes with collision failure. The save of dead points refers to whether a node is dead or not and whether it has reached the number of extensions. Determining whether this node can be extended by judging the size of the number of extensions. The save of child nodes with collision failure refers to the saved of new nodes that will collide, so as to ensure the search space of a node. In Fig. 3(d), the black circle is an obstacle. Node g has expanded four child nodes a, b, c, and d. If it continues to expand, then new child nodes (blue node e) will inevitably be generated in the repeat angle. This extension will fail due to obstacles. If we do not save the e node, the new sampling

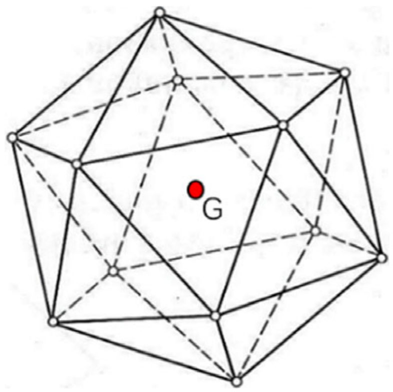


Figure 4. *Sparse nodes in three-dimensional space.*

point may be within the range of the repeat angle when sampling again, which will lead to redundant collision detection. Therefore, we saved the blue child node e which failed in collision detection, so as to avoid multiple sampling in the area of repeat angle. The sparse dead point saved strategy can not only guarantee the sparsity of nodes but also guarantee the completeness of the search space. In addition, the most important point is that the algorithm will not enter the minimum state.

3.2. Three-dimensional sparse dead point saved strategy

Similar to 2D space, in 3D space, for a parent node (the center G of the regular icosahedron in Fig. 4), it can extend outward at most 12 child nodes, and the distance between each child node should be greater than or equal to the extended step size (where the extended step size is equal to the edge length ‘a’ of the regular icosahedron), so as to ensure the sparseness of the extended tree. However, the distance from the center of the icosahedron to each vertex is not ‘a’, but $\frac{\sqrt{10+2\sqrt{5}}}{4}$a, which is about 0.95a. Since the distance between each node in the tree is almost impossible to be equal to the extended step size, we still take the extended step size and the threshold of node distance as the edge length of the positive icosahedron.

If a node has been collision detected more than 12 times, it is a dead point and will not be extended. For the starting point, its number of expansion is 12, and for the rest nodes, their number of expansion is 11. Nodes whose collision detection times exceed their respective number of expansion will no longer have opportunities to expand and become dead points. In 3D space, we can greatly reduce the number of collision detection and speed up the search speed of random tree by using dead point saved strategy.

4. The algorithm simulation

4.1. Two-dimensional and three-dimensional environment

We set up 2D and 3D environments as shown in Fig. 5, and each environment is divided into complex and simple situations. MATLAB 2020B is used on Windows 10 system to simulate RRT, SDPS-RRT, RRTConnect, SDPS-RRTConnect algorithms in different environments. The system uses Intel Core i7-10750H 2.6GHz CPU and 16GB RAM.

4.2. Algorithm simulation experiment

The algorithm is simulated in 2D and 3D environments. The average number of collision detection and the average consumption time of 500 times run by the four algorithms in 2D and 3D environments are shown in Table I.

Table I. Comparison of algorithms in 2D and 3D environment.

algorithm	RRT	SDPS-RRT	RRTconnect	SDPS-RRTconnect
Nocd(2D complex)	2516.3	472.48	1101.446	353.876
time(2D complex)	1.4356	0.30477	0.5943	0.2148
success rate(2D complex)	0.924.	0.968	0.936	1
Nocd(2D simple)	460.04	398.194	64.698	65.028
time(2D simple)	0.2583	0.23734	0.0361	0.0357
success rate(2D simple)	1	1	1	1
Nocd(3D complex)	1053.866	410.25	792.52	369.688
time(3D complex)	1.7579	0.73965	1.2766	0.67747
success rate(3D complex)	0.876	0.954	0.908	0.992
Nocd(3D simple)	709.17	426.29	91.512	66.354
time(3D simple)	0.78729	0.49669	0.10338	0.076594
success rate(3D simple)	1	1	1	1

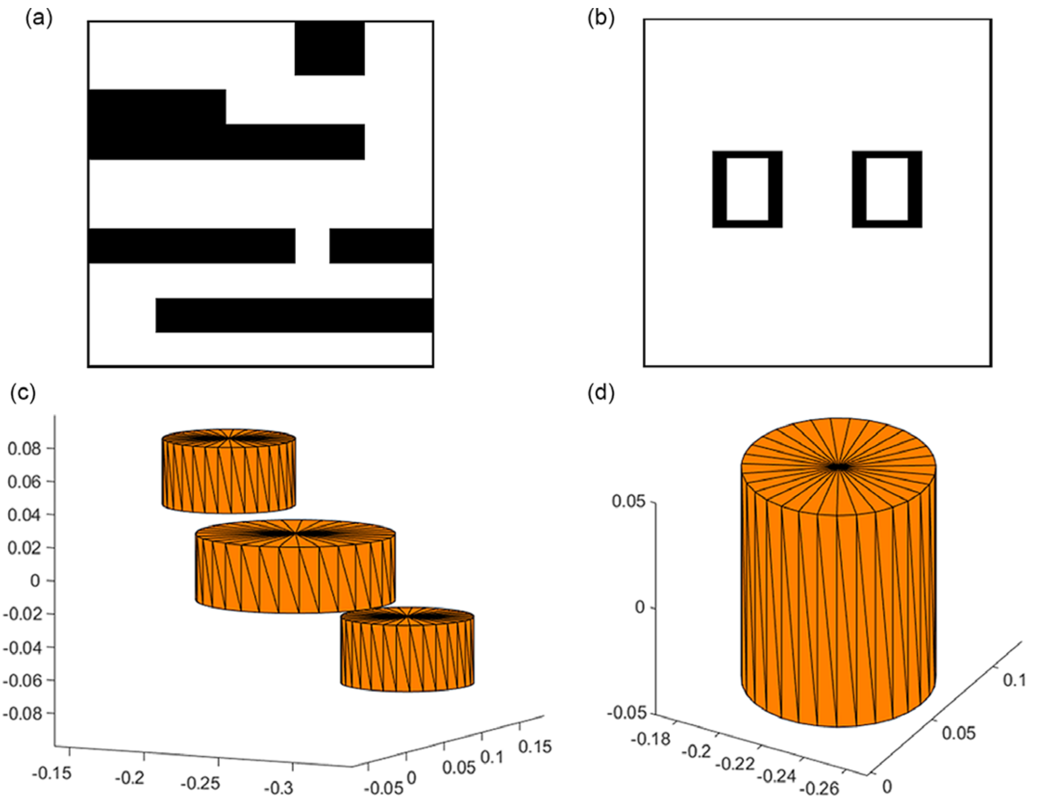
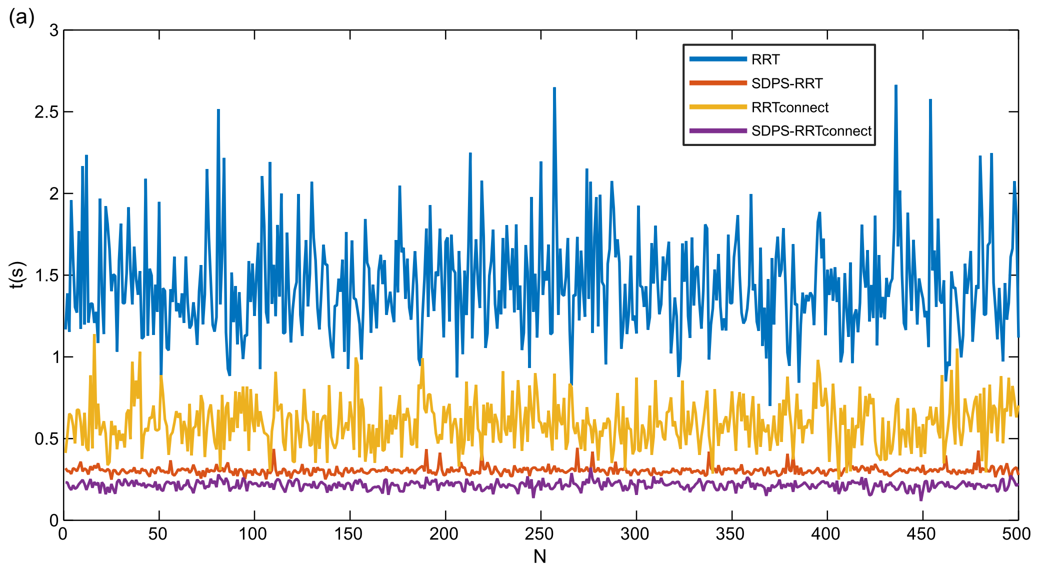


Figure 5. *Simulation environment. (a) 2D complex environment (b) 2D simple environment (c) 3D complex environment (d) 3D simple environment.*

Nocd in the table represents the number of collision detection times. In the complex environments, RRT algorithm takes the longest time to find the path after multiple collision detection. The collision detection times of SDPS-RRT algorithm are greatly reduced, which is about 35% of that of RRT algorithm, and the time is also reduced. In addition, the collision detection times of RRTConnect algorithm are about twice that of SDPS-RRT algorithm, and it consumes more time than SDPS-RRT algorithm. SDPS-RRTConnect algorithm has the least collision detection times, but it is only a little



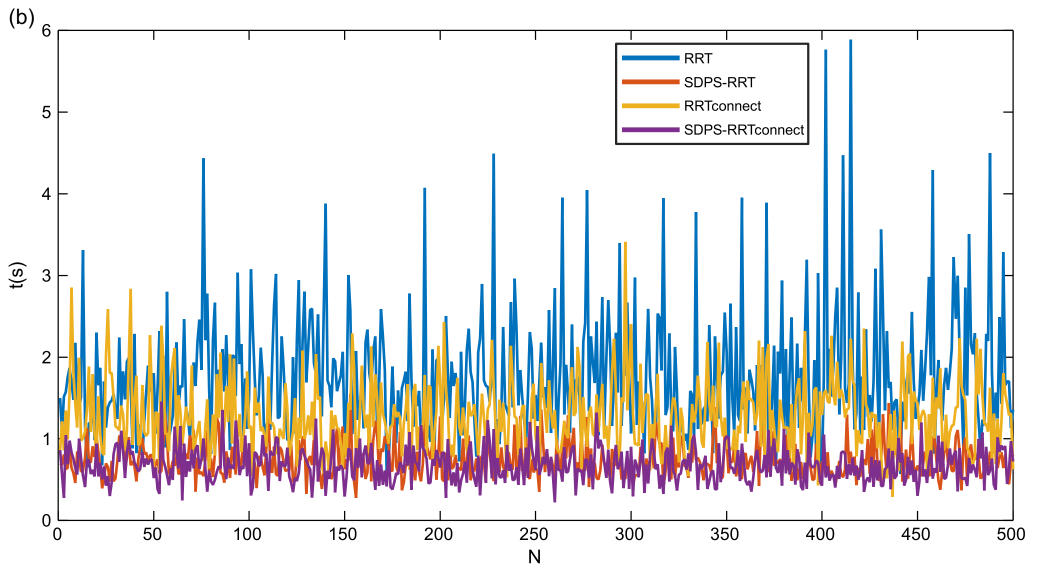


Figure 6. *Time taken for the four algorithms to run 500 times in 2D and 3D complex environment. (a) 2D complex environment (b) 3D complex environment.*

less than SDPS-RRT algorithm, which is about 40% of the collision detection times of RRTConnect algorithm. In complex environment, the improved algorithm can greatly reduce the times of collision detection and speed up exploration. The search speed of the ordinary RRTConnect algorithm is 27% faster than that of the ordinary RRT algorithm, and the SDPS-RRTConnect algorithm is 8% faster than the SDPS-RRT algorithm. It is shown that the double tree greedy link strategy plays a part role in the three-dimensional complex environment without improvement, while the effect becomes smaller after improvement. Compared with the unimproved algorithm, the improved algorithm can speed up the searching speed by about 50% in the complex environment. It is worth noting that the SDPS-RRT algorithm can search faster than the ordinary RRTConnect algorithm, which further proves the effectiveness of the dead point save strategy in complex environments. In a complex environment, the double tree greedy connection strategy plays a smaller role than the dead point save strategy proposed in this paper. This is because in a complex environment, there are many obstacles, and the nodes in the tree will

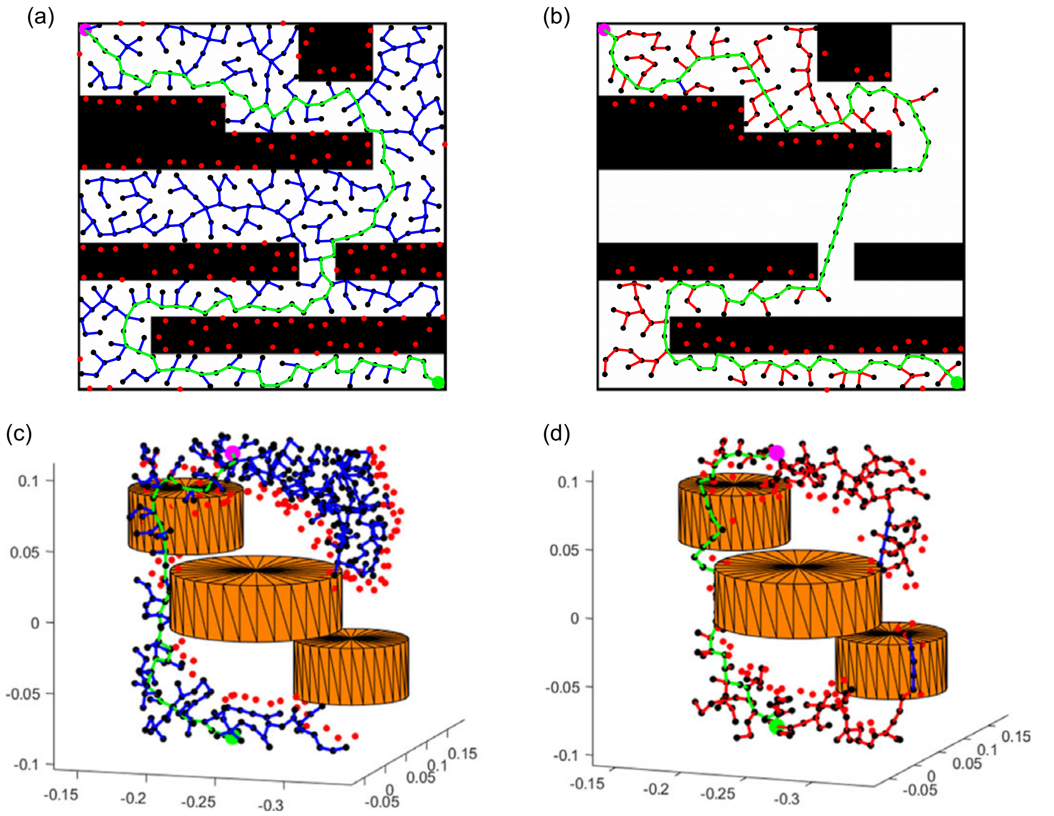


Figure 7. *Effect of SDPS algorithm in 2D and 3D environment. (a), (b) are the effects of simple environment and (c), (d) are the effects of complex environment.*

encounter obstacles if they expand the step size once or twice. Therefore, the greedy link strategy cannot play a good role, but will cause more collision detection. If the random sampling point always collides with the nearest neighbor node, the RRT algorithm will consume a lot of time and even fall into the minimum situation. The dead point save strategy avoids the repeated detection of collision prone nodes, ensures the search space, reduces the times of collision detection, and accelerates the expansion speed of random tree. In the complex environment, the success rate of RRT algorithm is the lowest, and the success rate of RRTConnect algorithm is slightly higher. The success rate of the improved algorithm has been increased to a certain extent, and the success rate of SPDS-RRTConnect algorithm is the highest.

In the simple environments, the collision detection times of the two RRTConnect algorithms were reduced by about 87% compared to the collision detection times of the two RRT algorithms. Correspondently, the two RRTConnect algorithms consumed approximately 85% less time than the normal RRT algorithm. In simple environments, the double tree greedy connection strategy plays a more important role than the dead point save strategy. This is because in simple environments, the greedy connection strategy can speed up the expansion of the extension tree. There are more paths from the beginning to the end. For nodes in the tree, the path can be easily found without repeated collision detection due to fewer obstacles. In other words, the extended tree can find the path without searching most of the workspace or even the whole workspace. The purpose of the dead point save strategy is to ensure that nodes in the tree can explore other directions if they encounter obstacles in one direction. The reduced probability of encountering obstacles naturally makes the dead point save strategy less effective. Because most nodes in simple environments are not dead points. Because of the simplicity of the environment, all the algorithms have better results and can be successful.

Table II. *Angle of each joint.*

	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
Start pose	−0.0903	1.0424	0.8399	1.1608	1.2276	0.5686
Goal pose	−0.3215	0	0	1.8055	0	1.4840

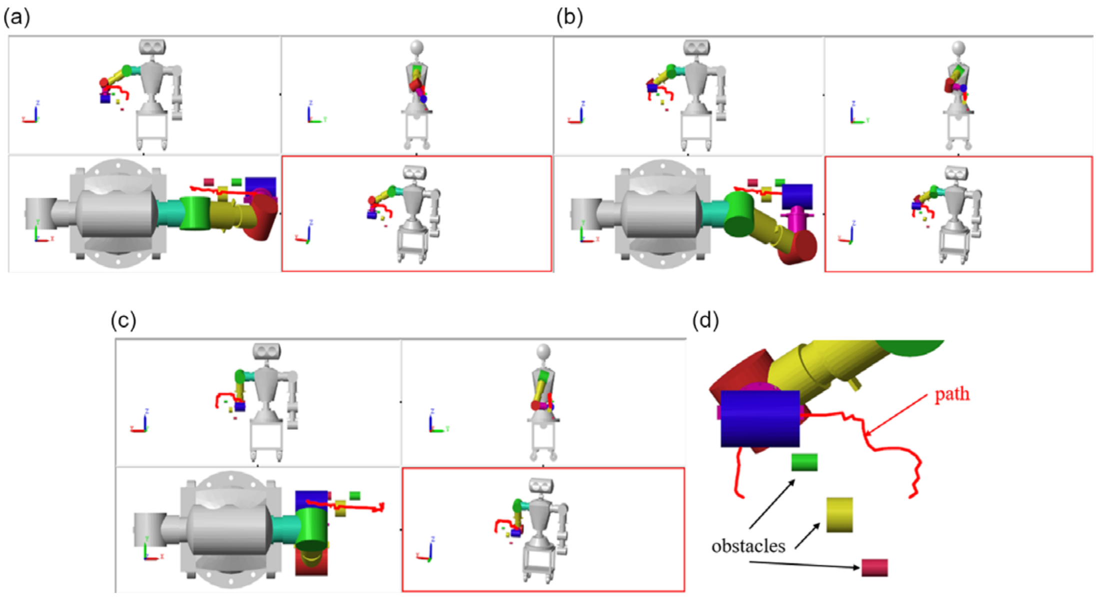


Figure 8. *Motion process of manipulator arm. (a) The SDPS – RRT algorithm (b) The SDPS – RRTconnect algorithm*

As can be seen from Fig. 6, RRT algorithm and RRTConnect algorithm have poor stability and long time consumption, while SDPS-RRT algorithm and SDPS-RRTConnect algorithm have strong stability and short time consumption. The effect of SDPS-RRT and SDPS-RRTConnect algorithms for complex environments is shown in Fig. 7.

The red points in Fig. 7 are collision points. Compared with Fig. 1, the nodes extended by the SDPS algorithm are more sparse and a large number of invalid searches are removed. When obstacles are encountered, SDPS algorithm will not continue to explore after certain collision detection, avoiding local minimum value and greatly speeding up the search speed of random tree.

To sum up, improved algorithm can achieve good results in both simple environments and complex environments. In simple environments, greedy link strategy plays a major role and double tree fast expansion of connection. In complex environment, dead point saved strategy plays a major role in avoiding repeated searches and ensuring search speed.

5. Experiment of Manipulator Motion Planning

5.1. Simulation of manipulator motion planning

In this paper, the laboratory self-made robot is taken as the research object, and the robot simulation environment is built in MATLAB. Obstacles and manipulator poses are set, in which the initial pose and target pose (unit: m) of the end of the manipulator are

$$P_{start} = \begin{bmatrix} 1 & 0 & 0 & 0.06 \\ 0 & -1 & 0 & -0.22 \\ 0 & 0 & -1 & 0.19 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

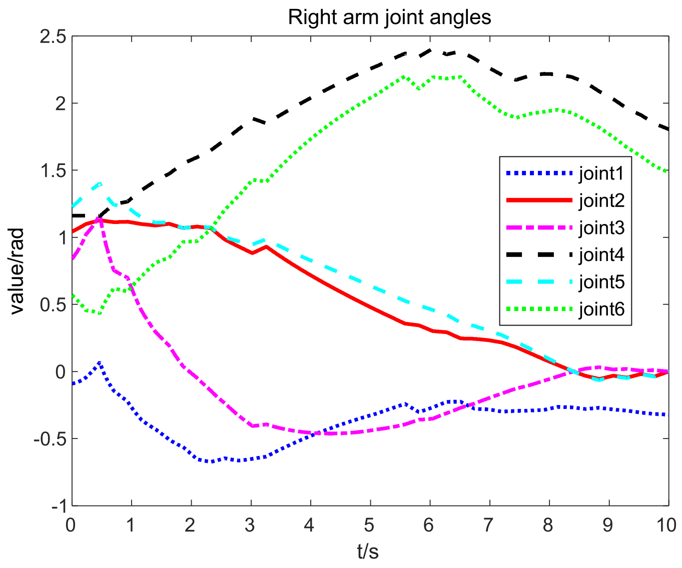


Figure 9. *Angle of each joint of the manipulator. (a) t = 0s, (b) t = 3s, (c) t = 10s, (d) local amplification at t = 3s.*

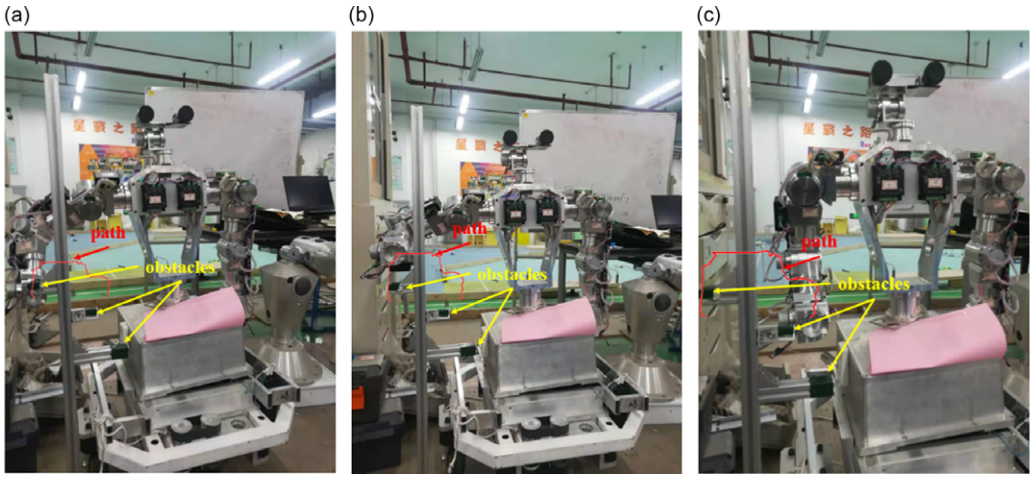


Figure 10. *The position of the manipulator at t = 0, 3, and 10s. We marked the obstacles with blue tape. As can be seen from Fig. 10, the manipulator avoided obstacles in the process of movement. The effectiveness of the algorithm is demonstrated.*

$$P_{end} = \begin{bmatrix} 1 & 0 & 0 & 0.06 \\ 0 & -1 & 0 & -0.22 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

The joint angles corresponding to the position and pose are shown in Table II (unit: rad).

The path planned by SDPS-RRTConnect algorithm in a complex environment was simulated. We did not shorten or optimize the path to verify the authenticity of the path planned by the algorithm. The motion process of the manipulator is shown in Fig. 8.

Fig. 8 shows the position of the manipulator at $t = 0$s, 3s, and 10s. There are three obstacles in the environment, red, green, and yellow, and the red curve is the path planned by SDPS-RTConnect algorithm. It can be seen from Fig. 8 that the manipulator avoids obstacles in the process of movement. Angle changes of each joint in the process of movement are shown in Fig. 9. The joint angles of the robot are continuous, which indicates that the path planned by SDPS-RRTConnect algorithm in this paper is feasible.

5.2. Real manipulator motion planning experiment

Experimental verification was carried out with a real manipulator, as shown in Fig. 10.

6. Conclusion

This paper puts forward some improvement measures for the shortcomings of RRT and RRTConnect algorithms. First, a sparse dead point saved strategy in two-dimensional environment is proposed. Simulation of two-dimensional environment shows that the algorithm can greatly reduce the number of collision detection, improve the planning speed, and avoid the phenomenon of local minimum. Then, the dead-point saved strategy is extended to three-dimensional environment, and the excellent effect of sparse dead-point saved strategy in complex environment is proved again through the simulation of three-dimensional environment. Finally, the simulation is carried out in MATLAB, and the simulation results show that the path planned by SDPS-RRTConnect algorithm can avoid obstacles. On this basis, a real robot experiment is carried out, and the experimental results show the effectiveness of SDPS-RRTConnect algorithm. However, optimality of path and dynamic obstacle avoidance are not considered in this paper. Future work includes trajectory planning and the application of this strategy to dynamic obstacle avoidance.

Acknowledgements. This work was supported by advance research projects in the field of manned spaceflight (No. M18GY300021).

Declaration of Conflicting Interests. The authors declare that there is no conflict of interest.

References

1. M. Kim, J. Kim, D. Shin and M. Jin, "Robot-based Shoe Manufacturing System," In Proceedings of the 18th International Conference on Control, Automation and Systems (ICCAS), Daegwallyeong, Korea, 1491–1494 (2018).
2. K. Junge, J. Hughes, T. G. -uruthel and F. Iida, "Improving robotic cooking using batch bayesian optimization," *IEEE Robot. Autom. Lett.* **5**(2), 760–765 (2020).
3. Y. Huang, Y. Zheng, N. Wang, J. Ota and X. Zhang, "Peg-in-hole assembly based on master-slave coordination for a compliant dual-arm robot," *Assem. Autom.* **40**(2), 189–198 (2020).
4. Z. Wang and X. Xiang, "Improved Astar Algorithm for Path Planning of Marine Robot. 2018 37th Chinese Control Cpnference(CCC)," 5410–5414 (2018).
5. D. D. Zhu and J. Q. Sun, "A new algorithm based on dijkstra for vehicle path planning considering intersection attribute," IEEE Access, 19761–19775 (2021).
6. O. R. Ulises, M. Oscar and S. Roberto, "Mobile robot path planning using membrane evolutionary artificial potential field," *Appl. Soft Comput.* **77**, 236–251 (2019).
7. A. C. Marco, A. R. Victor and H. B. Uriel, "Mobile robot path planning using artificial bee colony and evolutionary programming," *Appl. Soft Comput.* **30**, 319–328 (2015).
8. I. B. Jeong, S. J. Lee, J. H. Kim and Quick-RRT*, "Triangular inequality-based implementation of RRT* with improved initial solution and convergence rate," *Expert Syst. Appl.* **123**, 82–90 (2019).
9. S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.* **30**, 846–894 (2011).
10. J. Nasir, F. Islam, U. Malik, Y. Ayaz, O. Hasan, M. Khan and M. S. Muhammad, "RRT*-SMART: A rapid convergence implementation of RRT," *Int. J. Adv. Robot. Syst.* **10**, 299 (2013).
11. N. Iram, K. Amna and H. Zulfiqar, "A Comparison of RRT, RRT* and RRT*-Smart Path Planning Algorithms," *Int. J. Comput. Sci. Net. Secur.* **16**(10), 20–27 (2016).

12. M. V. Weghe, D. Ferguson and S. S. Srinivasa, “Randomized path planning for redundant manipulators withoutinverse kinematics,” In Proceedings of the 2007 7th IEEE-RAS International Conference on Humanoid Robots,Pittsburgh, PA, USA, 29 November–1 December, 477–482 (2007).
13. S. M. Lavalle and J. J. Kuffner, “Rapidly-Exploring Random Trees: Progress and Prospects,” In Algorithmic & Computational Robotics New Directions; CRC Press: Boca Raton, FL, USA, 293–308 (2000).
14. J. J. Kuffner and S. M. Lavalle, “RRT-connect: An efficient approach to single-query path planning,” In Proceedingsof the IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, 24–28 April, 995–1001 (2020).
15. S. M. LaValle and J. J. Kuffner, Jr, “Randomized kinodynamic planning,” *Int. J. Robot. Res.* **20**, 378–400 (2001).
16. C. Yuan, W. Zhang, G. Liu, X. Pan and X. Liu, “A heuristic rapidly-exploring random trees method for manipulator motion planning,” *IEEE Access.* **8**, 900–910 (2020).
17. X. Wang, X. Luo, B. Han, Y. Chen, G. Liang and K. Zheng, “Collision-free path planning method for robots based on an improved rapidly-exploring random tree algorithm,” *Appl. Sci.* **10**, 1381 (2020).
18. K. Wei and B. Y. Ren, “A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved RRT algorithm,” *Sensors*, **18**(2), 571, (2018).
19. X. Y. Wang, “Bidirectional Potential Guided RRT* for Motion Planning,” *IEEE Access.* **7**, 95034–95045, (2019).
20. H. J. Zhang, “Path Planning of Industrial Robot Based on Improved RRT Algorithm in Complex Environments,” *IEEE Access.* **6**, 53296–53306, (2018).
21. G. Kang, Y. B. Kim, Y. H. Lee, H. S. Oh, W. S. You and H. R. Choi, “Sampling-based motion planning of manipulator with goal-oriented sampling,” *Intell. Serv. Robot.* **12**, 265–273 (2019).
22. W. Qiao, Z. Fang and B. Si, “A sampling-based multi-tree fusion algorithm for frontier detection,” *Int. J. Adv. Robot. Syst.* **16**(4), 1729881419865427 (2019).

Cite this article: J. Xu and J. Wang (2022). “Effective motion planning of manipulator based on SDPS-RRTConnect”, Robotica **40**, 1855–1867. https://doi.org/10.1017/S0263574721001417