# Smooth-RRT*: An Improved Motion Planner for Underwater Robot


1st Kehao Wang
*School of Information Engineering*
*Wuhan University of Technology*
Wuhan, China
kehao.wang@whut.edu.cn

2nd Shuaifu Li
*School of Information Engineering*
*Wuhan University of Technology*
Wuhan, China
lishuaifu@whut.edu.cn

3rd Yang Wang
*National Engineering Research Center*
*for Water Transport Safety*
*Wuhan University of Technology*
Wuhan, China
wangyang.itsc@whut.edu.cn

4th Jing Xi
*The First Military Representative Office*
*of the First Military Representative Bureau*
*of the Army Armaments Department in Beijing,PLA*
Beijing, China
jingfan1018@163.com



***Abstract*—In underwater search and rescue, it is very important for underwater robot to reach the rescue position quickly. Planning path in advance is very important to save rescue time and energy consumption. Therefore, it is meaningful to find a better and shorter path as soon as possible. As a common method of path planning , RRT* has the disadvantages of high cost and slow convergence. To solve these flaws, an improved motion planner for underwater robots is proposed in this paper. In this study, the simulation experiments were divided into two-dimensional conditions and three-dimensional conditions, where used point cloud of real underwater scene to find a better initial solution. Based on RRT*, this paper finds the ancestor node farthest from the sampling point and without collision in the random tree as parent node, adds intermediate nodes in the path according to the step size, and uses trigonometric inequality many times throughout the process, so as to obtain an optimized path. Through a large number of simulation experiments, the results show that the cost of path is less and the convergence speed is faster than RRT* and Q-RRT*.**



***Index Terms*—path planning, Rapidly-exploring random tree (RRT), path optimization, robot, sampling-based algorithms**


## I. Introduction

In the scene of underwater, the use of underwater robots is an important means. Taking emergency search and rescue as an example, in the shipwreck of the Oriental Star of the Yangtze River in 2015, to rescue the survivors in the overturned hull, robots need to enter the sunken ship for detection and rescue. Another example is the trapped incident of the Thailand youth football team in 2018. Due to the tortuous and narrow path , it took rescuers many days to find trapped people. To improve the efficiency of underwater operation, path planning has received extensive attention.

Generally, path planning is to find a continuous path without collision connecting start state and end state according to criteria of shortest distance and time optimization. Currently, common path planning algorithms include Dijkstra [1], A-star (A*) [2], Artificial Potential Field (APF) [3], Probabilistic Roadmap (PRM) [4], Rapidly-exploring Random Trees (RRT) [5], and so on.

The Dijkstra compares various cases by traversing all nodes. It has good robustness, but low efficiency [6]. A-star algorithm adds heuristic search information compared with Dijkstra. Although it speeds up the search efficiency, it is still not suitable for high-dimensional space. The APF is a virtual force method that simulates the motion of an object under the combined action of attraction and repulsion, but it easily falls into a local optimal solution [7]. The PRM is a sampling-based algorithm that has good expressiveness in a high-dimensional environment, but it is completely probabilistic and not optimal [8]. Compared with these algorithms, RRT is a sampling based algorithm, which depends on random sampling and collision detection. RRT can generate a feasible path quickly without establishing an environment model and is suitable for high-dimensional situations [9]. However, the initial solution of RRT is tortuous and cannot guarantee optimality [10].

RRT* [11] is proposed as an excellent variant of RRT to obtain an optimized path by selecting a better parent node and rewiring nearby nodes. The algorithm has probability completeness and asymptotic optimality [12]. RRT* is a milestone and has become the basis of subsequent research work [13]. To improve the performance of the algorithm, many scholars had proposed some improved algorithms. Jhang and F. Lian [14] proposed the bidirectional RRT* algorithm (B-RRT*), which can improve the convergence speed and use the admissible heuristic method to selectively generate new nodes to improve the path. The RRT*-smart [15] proposed by Nasir et al. has two main characteristics: intelligent sampling and path optimization. After the initial path is generated, the trigonometric inequality principle is used to crop out redundant nodes. When optimizing the generated path, it relies on the generation of beacon nodes to reduce the path cost.

In addition, reducing path nodes can also speed up con-

vergence speed and shorten the path length. Jeong et al. [16] proposed Q-RRT*, which enlarges the set of possible parent nodes by considering not only the nearby nodes as in RRT* but also their ancestor nodes. The results show that the generation path is better than RRT.

In general, scholars have made great efforts to improve RRT*. Because the mechanism of selecting parent nodes has not been changed, there are still some problems, such as path redundancy, slow convergence and low efficiency. To solve these problems, this paper proposes Smooth-RRT* (S-RRT*) to further improve the performance The proposed algorithm further improves the performance of initial solution quality and convergence speed. The contributions of this paper are as follows.

- Based on the idea of backtracking, the farthest ancestor node is defined as the parent node, which ensures the optimal temporary cost. It saves more time than selecting the parent node in a vertex set of a selected space as in RRT*.
- We add intermediate nodes in the unit of step size between each node and its parent node, which maintains the advantage of small step size to obtain a better solution.
- Experiments show that our algorithm has more advantages for initial solution and convergence speed.

## II. Background

This part formalizes the path planning problem and analyzes the primitive modules of RRT*.

### A. Problem definition

Let $S = (0,1)^d$ define the given workspace, where $d$ is the dimension of workspace. Let $S_{oct} \subset S$ be the obstacle region and $S_{free} = S/S_{oct}$ define the free region. Given starting point $X_{start}$ and goal point $X_{goal}$ , then $\{S, X_{start}, X_{goal}\}$ defines a problem of path planning. A path $\sigma : [0,1] \mapsto S$ has bounded variation, it is continuous. A path $\sigma(\tau) \in S_{free}$ for all $\tau \in [0,1]$ , it is collision-free.

Furthermore, a problem of path planning is to find a continuous path without collision $\sigma : [0,1] \mapsto S_{free}$. To solve this problem, this paper builds a Path-Tree $T = (V, E)$, where $V$ represents the set of path points in Path-Tree and $E$ represents the set of edges that connect points in $V$. That is to say, the key of this paper is to find a path connecting starting state and end state, by finding an ordered set of points that avoids obstacles in a given space, and connecting these points with straight lines.

Definition 1: Feasible initial solution. It refers to find a collision-free path $\sigma : [0,1] \mapsto S_{free}$. And $\sigma(0) = X_{start}$, $\sigma(1) = X_{goal}$.

Definition 2: Optimal solution. It refers to find a collision-free path $\sigma^* \subset \sigma$ that minimizes the cost $Cost(\sigma^*)$ such that $Cost(\sigma^*) = min\{Cost(\sigma)\}$, where $Cost(\sigma)$ is the distance to reach to $X_{goal}$ along with a path $\sigma$.

when $d = 3$, assume that the vertex set of the final path is defined as $P$, $P = \{p_1, p_2, ..., p_N\}$, where N is the number of path points. Let $\| p_2 - p_1 \|$ represent the Euclidean distance between $p_1$ and $p_2$. Therefore, our proposed optimization problem can be expressed as

$$\begin{aligned} \min_{P} \quad & \sum_{i=2}^{N} \| p_i - p_{i-1} \| \\ \text{s.t.} \quad & C_1 : N \leq N_{max} \\ & C_2 : p_1 = X_{start}, , p_N = X_{goal} \\ & C_3 : p_i \in S_{free} \end{aligned} \tag{1}$$

In the above formula, the goal of the optimization problem is to minimize the total cost of the final path. $C_1$ indicates that the number of path points should be less than a given maximum value $N_{max}$. $C_2$ indicates that the final path should connect the starting point and the goal point. $C_3$ indicates that the final path is collision-free.

### B. RRT*

RRT is a classical algorithm suitable for multi-dimensional space. It can effectively solve problems such as complex constraints in multidimensional spaces by avoiding spatial modeling. It takes starting point as the root node and selects leaf nodes by random sampling to generate a random tree. If the leaf node reaches target point or the distance from target point is less than the given threshold, a continuous path connecting start point and end point will be found in the random tree. As an excellent variant of RRT, RRT* ensures asymptotic optimality, that is, with the increase in the number of iterations, it will converge to the optimal solution. RRT* improves performance by choosing the parent node and rewiring the nearby nodes based on RRT, to significantly reduce the cost of path.

## III. Proposed

This section proposes the S-RRT* algorithm based on RRT*, which can further reduce the cost from $X_{start}$ to $X_{goal}$. Two main modules of improvement are introduced in detail, $ChooseParent$ based on backtracking and $AddNodes$ based on step size.

### A. Proposed algorithm

The S-RRT* in this paper explores space such as RRT*. In RRT*, the method of selecting parent node for a new node is to calculate the overall cost within a certain radius, and take the node with the lowest cost as the parent node of the new node. Therefore, the selection of radius is very important. The larger the radius is, the better the initial solution that will be obtained, but the calculation time will also increase because of the doubling of the number of adjacent points.

To eliminate the influence of the radius element on calculation events, this paper abandons the radius parameter rather than choosing the farthest ancestor node without collision as the parent node.

In addition, the path finally generated by RRT* contains few nodes and has a large step size. We found that a large step size is not conducive to the generation a optimal path. Therefore, once the parent node of the new node is determined, a series

of intermediate nodes will be created. The pseudo-code of S-RRT* is shown in Algorithm 1.

**Algorithm 1:** S-RRT*

**input :** the start point $X_{start}$
the goal point $X_{goal}$
the workspace $S$
the number of the iterations $n$
**output:** $T$

V $\leftarrow \{X_{start}\}$; E $\leftarrow \emptyset$
T $\leftarrow (V, E)$
**for** $i \leftarrow 1$ **to** $n$ **do**
  $X_{rand} \leftarrow$ `SampleFree(`$i$`)`
  $X_{nearest} \leftarrow$ `GetNearest(`$T$, $X_{rand}$`)`
  $(X_{new}, \sigma) \leftarrow$ `Steer(`$X_{rand}, X_{neareat}, q$`)`
  **if** `CollisionFree(`$X_{new}$, $X_{nearest}$`)` **then**
    $X_{parent} \leftarrow$ `ChooseParent(`$T$, $X_{nearest}$, $X_{new}$`)`
    V $\leftarrow V \bigcup \{X_{new}\}$
    F $\leftarrow$ `AddNodes(`$X_{new}$, $X_{parent}$`)`
    E $\leftarrow E \bigcup F$
  **end**
  **if** $\|X_{new} - X_{goal}\| \leq q$ **then**
    V $\leftarrow V \bigcup \{X_{goal}\}$
  **end**
**end**
Return $T$

The primitive modules of S-RRT* is described as follows.

- $SampleFree$: It returns a randomly sampled state $X_{rand}$ in $S_{free}$.
- $GetNearest$: It returns the vertex $X_{nearest}$ closest to $X_{rand}$ in $T$. And it is determined by Euclidean distance.
- $Steer$: It returns a path connecting $X_{nearest}$ and $X_{nearest}$ at the step size $q$.

### B. ChooseParent

Inspired by RRT*, using the triangular inequality criterion multiple times can further reduce the cost of the solution. The process of $ChooseParent$ in S-RRT* is to find the reachable node $X_{parent}$ from the ancestors of $X_{nearest}$. The node will become a candidate parent node of $x_{new}$. Based on trigonometric inequality, the cost of distance connecting $X_{parent}$ and $X_{new}$ will make the cost from $X_{start}$ to $X_{new}$ lower than connecting $X_{nearest}$ and $X_{new}$. It searches only the ancestors of $X_{nearest}$, not the nodes around $X_{new}$. Different from RRT*, fewer search nodes can significantly reduce the computing time. The function $Parent$ returns the parent node of the current point. The pseudo-code of $ChooseParent$ is shown in Algorithm 2.

### C. AddNodes

After observing the path generated by RRT*, we think that the connection between $X_{new}$ and $X_{start}$ can be further optimized by changing the direction in advance near the obstacles. Based on trigonometric inequality, the cost of path will be significantly reduced. Due to the lack of nodes in the path, we add intermediate nodes aimed changing the direction in advance. The node used to optimize the path is created in the $AddNodes$ process. During the period, intermediate nodes are created based on the initial step size $q$. This method effectively reduces the cost of initial solution by maintaining a fixed step size and inherits the advantage of a small step size which results in a smoother path. The pseudo-code of $AddNodes$ is shown in Algorithm 3.

**Algorithm 2:** ChooseParent

**input :** the Path-Tree $T$
the new node $X_{new}$
the nearest node $X_{nearest}$
**output:** $X_{parent}$

$X_{parent} \leftarrow X_{nearest}$
**while** $X_{parent} \neq X_{start}$ **do**
  $X_{temp} \leftarrow Parent(X_{parent})$
  **if** `CollisionFree(`$X_{new}$, $X_{temp}$`)` **then**
    $X_{parent} \leftarrow X_{temp}$
  **else**
    Return $X_{parent}$
  **end**
**end**
Return $X_{parent}$

**Algorithm 3:** AddNodes

**input :** the new node $X_{new}$
the parent node $X_{parent}$
the step size $q$
**output:** $F$

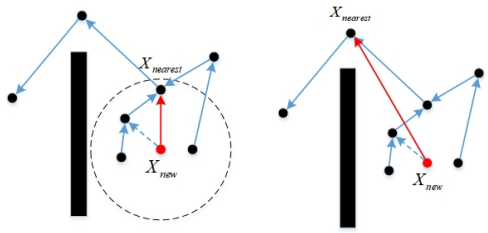
Dis = `Distance(`$X_{new}$, $X_{parent}$`)`
Num = Dis / q
Dir = ($X_{new}$-$X_{parent}$) / Dis
$X_{temp} \leftarrow X_{parent}$
**for** $i \leftarrow 1$ **to** $Num$ **do**
  $X_{node} \leftarrow X_{parent} + i * q * Dir$
  F $\leftarrow F \bigcup \{(X_{node}, X_{temp})\}$
  $X_{temp} \leftarrow X_{node}$
**end**
Return $F$

### D. Analysis

Compared with RRT*, the algorithm proposed in this paper improves performance by choosing the parent node and adds intermediate nodes to path tree, which can reduce the cost of the path. In RRT*, the method of selecting the parent node is to first place all nodes within the range with $X_{new}$ as the center and radius $r$ into set $U$, then successively calculate the current cost of each node as the parent node of $X_{new}$, and take the node with the lowest cost as the parent node of $X_{new}$. Obviously, the choice of radius parameter $r$ has a great

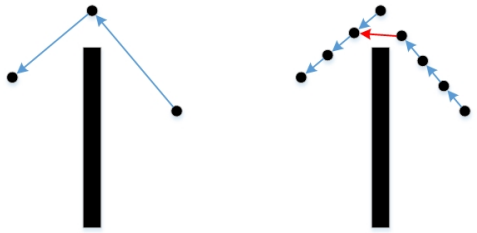
influence on the final result. When $r$ is larger, the more parent nodes available for $X_{new}$ to choose, the greater the possibility of obtaining a better path, but the resulting time cost also rises sharply.

Different from RRT*, S-RRT* discards the radius parameter $r$ and takes the farthest ancestor node as the parent node of $X_{new}$ as shown in Fig. 1, which can greatly reduce the cost of the path and the time overhead. In addition, by observing the path generated by RRT*, there are many places to optimize the path by turning in advance. We can find that the distance between each node and the parent node is much larger than the preset step size. We know that small step size can usually generate a better path. Therefore, after choosing the parent node, this algorithm adds the intermediate node according to the step size, and the next node may find a better parent node. Fig. 2 depicts the process.




(a) (b)

Fig. 1. The process of ChooseParent. (a) RRT*. (b)S-RRT*.



(a) (b)

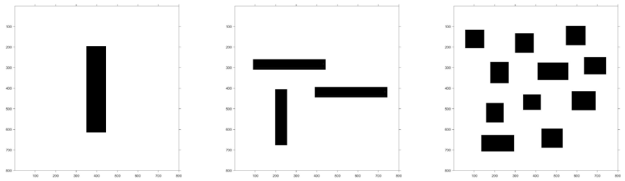
Fig. 2. The process of AddNodes. (a) RRT*. (b)S-RRT*.

## IV. Experiments and Discussion

In this section, we compared the performance of RRT*, Q-RRT* and S-RRT* in same environment. In the simulation, experiments were carried out in 2D and 3D environments, respectively. Where the 3D environment data is a 3D point cloud of an underwater scene collected by sonar, and obstacles are appropriately added for the objectivity of the experiment. The experiments were run on Intel i5-7300K CPU with 8G RAM.

### A. Experiment in 2D Environment

The 2D environment is shown in Fig. 3. There are three maps that represent simple, moderate and complex scenarios. The size of each map is $800 \times 800$, the black areas are obstacles. The comparative experiment includes two aspects. The first aspect is the quality of the initial solution generated by each algorithm. The second aspect is the convergence speed of each algorithm and the optimal solution obtained by convergence. The experimental result is the path generated by 2000 iterations.



(a) (b) (c)

Fig. 3. The environment of 2D. (a) Simple scenario. (b) Moderate scenario. (c) Complex scenario.

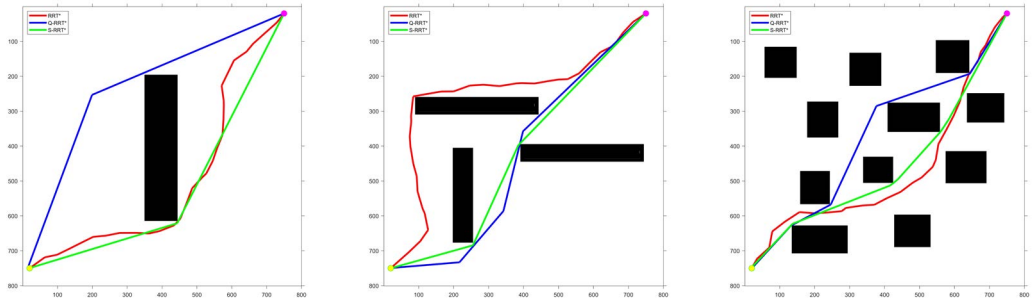
In Fig. 4, the purple point represents the starting point, the green point represents the goal point. As shown, all three algorithms generated a path connecting the starting point and the goal point. Experimental results verified the effectiveness of the algorithm in initial path generation. what can be seen is that, the path generated by RRT* is the longest, followed by Q-RRT* whose path avoids many bends although not optimal, the path generated by S-RRT* is the shortest due to closer proximity to obstacles.

To overcome the randomness of sampling, 500 independent experiments had been run for each algorithm. Table I shows the cost of the path of three algorithms. Compared with RRT* and Q-RRT*, the cost of S-RRT* is reduced by 9.39% and 1.61% in the simple scenario on average, reduced by 9.88% and 3.69% in the moderate scenario, and reduced by 7.19% and 2.54% in the complex scenario. These statistical data show that S-RRT* has more advantages in the quality of the initial solution.

TABLE I
The Comparison of Cost

| Scenario | Algorithm | Max | Min | Avg |
|---|---|---|---|---|
| Simple | RRT* | 1320.18 | 1155.57 | 1221.72 |
| | Q-RRT* | 1170.62 | 1107.28 | 1125.06 |
| | S-RRT* | 1118.40 | 1083.21 | 1106.94 |
| Moderate | RRT* | 1372.10 | 1124.27 | 1201.88 |
| | Q-RRT* | 1210.92 | 1084.36 | 1124.69 |
| | S-RRT* | 1101.27 | 1077.38 | 1083.10 |
| Complex | RRT* | 1368.11 | 1104.08 | 1162.06 |
| | Q-RRT* | 1238.59 | 1074.62 | 1106.57 |
| | S-RRT* | 1098.60 | 1069.76 | 1078.40 |

In Fig. 5, the convergence speed of the three algorithms is shown, which is the result of the complex scenario. With the increase in the number of iterations, the three algorithms finally converged to the best state. However, when S-RRT* iterated 1100 times, it produces the best path and keeps it, which is the fastest of the three algorithms. In addition, the path obtained by the convergence of this algorithm is also the



(a) (b) (c)

Fig. 4. Comparison of the quality of the initial solution in 2D environment. (a) Simple scenario. (b) Moderate scenario. (c) Complex scenario.

best. The result shows that S-RRT* has more advantages in convergence speed.

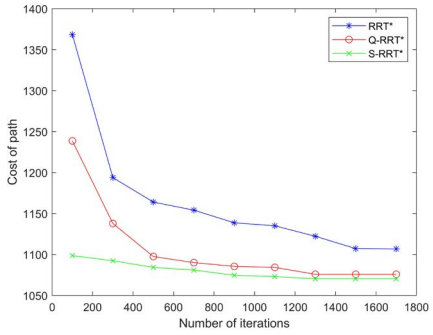



Fig. 5. Comparison of the convergence speed in 2D environment

### B. Experiment in 3D Environment

The environmental data is a 3D point cloud collected by sonar equipment in a pool. This point cloud presents a complex underwater three-dimensional environment, which contains 87104 points. The range of the X-axis is (-15, 12), the range of the Y-axis is (-3, 4), the range of the Z-axis is (-2, 4). The comparison of three algorithms in the three-dimensional environment is still obvious.

In Fig. 6, the purple point represents the starting point, the green point represents the goal point. We set two different groups of starting points and goal points to simulate two different scenarios. Obviously, all three algorithms generated a collision-free path connecting the starting point and the goal point. As shown in Fig. 6, the path generated by RRT* is the longest, followed by Q-RRT*. The path generated by our algorithm is the shortest and the final path is smoother. The result verifies the feasibility of the algorithm in three-dimensional environment.

To avoid contingency, we run the experiments of three algorithms independently 500 times in two scenarios. Table II shows the comparison of the cost of the path of three algorithms. The maximum, minimum and average values are listed. For average value, compared with RRT* and Q-RRT*. The path cost of S-RRT* is reduced by 12.7% and 6.05% in scenario1, reduced by 27.48% and 12.49% in scenario2.

TABLE II
THE COMPARISON OF COST

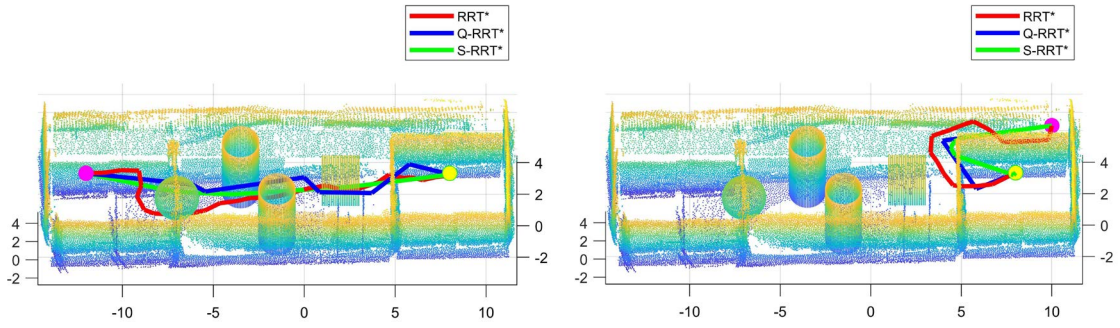
| Scenario | Algorithm | Max | Min | Avg |
|---|---|---|---|---|
| Scenario1 | RRT* | 27.72 | 22.40 | 24.55 |
| | Q-RRT* | 25.12 | 21.66 | 22.81 |
| | S-RRT* | 22.05 | 21.15 | 21.43 |
| Scenario2 | RRT* | 10.90 | 13.32 | 14.59 |
| | Q-RRT* | 13.98 | 10.52 | 12.09 |
| | S-RRT* | 12.35 | 10.01 | 10.58 |

Similarly, Fig. 7 shows the convergence speed of the three algorithms in scenario 1. It can be seen that the initial solution quality of S-RRT* is the highest, followed by Q-RRT* and RRT* is the worst. The result of S-RRT* algorithm has little fluctuation and converges to the optimal solution in 1200 iterations, while the convergence speed of the other two algorithms is slower, and the final result is not as good as S-RRT*. The statistical data verify the advantage of S-RRT* in convergence speed in three-dimensional environment.

### C. Discussion

From these experiments, the result shows that our algorithm performs better than RRT* and Q-RRT*. The advantages may come from the following reasons.

- Based on trigonometric inequality, the path generated by our algorithm will be significantly shorter and smoother. These superiorities are more prominent in complex environments
- Compared with RRT* and Q-RRT*, instead of the radius parameter, we find the farthest ancestor node for $X_{new}$, which makes it obtain the initial solution faster and converge faster.
- Creating intermediate nodes in steps between $X_{new}$ and $X_{parent}$ provides more choices for subsequent nodes to find the farthest parent node. Therefore, the cost of path will be further reduced.




(a) (b)

Fig. 6. Comparison of the quality of the initial solution in 3D environment. (a) scenario1. (b) scenario2.

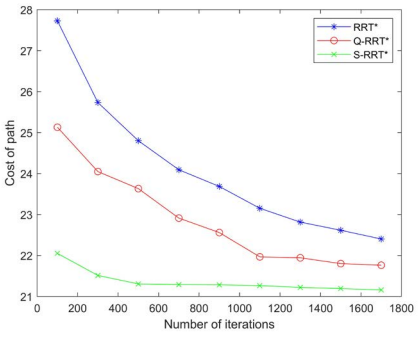



Fig. 7. Comparison of the convergence speed in 3D environment

## V. Conclusion

In this paper, an optimization algorithm called S-RRT* is proposed. It performs well in the quality of initial solution and convergence speed. The algorithm proposed in this paper inherits the advantages of RRT*, uses backtracking to select the parent node to minimize the current cost, and adds intermediate nodes to maintain the advantage of small step size, to provide a better choice for subsequent new nodes to select the parent. Experiments show that under the same conditions, the final path obtained by this algorithm is shorter and smoother than previous algorithms. In addition, we achieved good results in experimental simulation, and verified the path with ROV. Although, our work does not investigate the situation in dynamic environments. In addition, kinematic constraints are not considered in this paper, and we hope to continue to explore them in future work.

## Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grant 62172313

## References

[1] M. Luo, X. Hou and J. Yang, "Surface Optimal Path Planning Using an Extended Dijkstra Algorithm," in IEEE Access, vol. 8, pp. 147827-147838, 2020.

[2] R. Wen and M. Tong, "Mecanum wheels with Astar algorithm and fuzzy PID algorithm based on genetic algorithm," 2017 International Conference on Robotics and Automation Sciences (ICRAS), 2017, pp. 114-118.

[3] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," Proceedings. 1985 IEEE International Conference on Robotics and Automation, 1985, pp. 500-505.

[4] M. Baumann, S. Lonard, E. A. Croft and J. J. Little, "Path Planning for Improved Visibility Using a Probabilistic Road Map," in IEEE Transactions on Robotics, vol. 26, no. 1, pp. 195-200, Feb. 2010.

[5] H. Zhang, Y. Wang, J. Zheng and J. Yu, "Path Planning of Industrial Robot Based on Improved RRT Algorithm in Complex Environments," in IEEE Access, vol. 6, pp. 53296-53306, 2018.

[6] Y. Zhang, Y. Su, J. Yang, J. Ponce and H. Kong, "When Dijkstra Meets Vanishing Point: A Stereo Vision Approach for Road Detection," in IEEE Transactions on Image Processing, vol. 27, no. 5, pp. 2176-2188, May 2018.

[7] N. Zhang, Y. Zhang, C. Ma and B. Wang, "Path planning of six-DOF serial robots based on improved artificial potential field method," 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO), 2017, pp. 617-621.

[8] A. Francis et al., "Long-Range Indoor Navigation With PRM-RL," in IEEE Transactions on Robotics, vol. 36, no. 4, pp. 1115-1134, Aug. 2020.

[9] D. Ghosh, G. Nandakumar, K. Narayanan, V. Honkote and S. Sharma, "Kinematic Constraints Based Bi-directional RRT (KB-RRT) with Parameterized Trajectories for Robot Path Planning in Cluttered Environment," 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 8627-8633.

[10] R. Mashayekhi, M. Y. I. Idris, M. H. Anisi, I. Ahmedy and I. Ali, "Informed RRT*-Connect: An Asymptotically Optimal Single-Query Path Planning Method," in IEEE Access, vol. 8, pp. 19842-19852, 2020.

[11] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning with deterministic -calculus specifications," 2012 American Control Conference (ACC), 2012, pp. 735-742.

[12] W. Xinyu, L. Xiaojuan, G. Yong, S. Jiadong and W. Rui, "Bidirectional Potential Guided RRT* for Motion Planning," in IEEE Access, vol. 7, pp. 95046-95057, 2019.

[13] P. Pharpatara, B. Hriss and Y. Bestaoui, "3-D Trajectory Planning of Aerial Vehicles Using RRT*," in IEEE Transactions on Control Systems Technology, vol. 25, no. 3, pp. 1116-1123, May 2017.

[14] J. -H. Jhang and F. -L. Lian, "An Autonomous Parking System of Optimally Integrating Bidirectional Rapidly-Exploring Random Trees* and Parking-Oriented Model Predictive Control," in IEEE Access, vol. 8, pp. 163502-163523, 2020.

[15] F. Islam, J. Nasir, U. Malik, Y. Ayaz and O. Hasan, "RRT-Smart: Rapid convergence implementation of RRT towards optimal solution," 2012 IEEE International Conference on Mechatronics and Automation, 2012, pp. 1651-1656.

[16] In-Bae Jeong, Seung-Jae Lee, Jong-Hwan Kim, "Quick-RRT*: Triangular inequality-based implementation of RRT* with improved initial solution and convergence rate," Expert Systems with Applications, Volume 123, 2019,Pages 82-90.