Path Planning Method for Unmanned Surface Vehicle Based on RRT* and DWA

Xiaotian Zhang and Xiyuan Chen(✉)

School of Instrument Science and Engineering, Southeast University, Nanjing 210096, China
chxiyuan@seu.edu.cn

Abstract. Based on the optimized rapidly-exploring random tree (RRT*) and dynamic window approach (DWA), this paper proposes a method for path planning of unmanned surface vehicle. By inputting the global path generated by RRT* to the local path planner through soft constraints, and combining the obstacle information received by the sensor when the USV is working, the improved DWA is used to obtain a route that can be operated by the USV. The simulation results verify it can effectively avoid dynamic obstacles, ensure the safe navigation of USV in complex environments, and can meet real-time requirements.

Keywords: USV · Path planning · RRT* · DWA · Dynamic obstacle avoidance

1 Introduction

Unmanned surface vehicle (USV) is a type of vehicle carrying multiple types of sensors without manual control. Compared with manned vehicle, USV have the advantages of small dimension, full-time, low-cost, and have strong adaptability to complex water surfaces such as near shores or shallows, especially not suitable for manned vehicle to perform tasks [1]. Equipped with advanced control system, sensor system and communication system, it can perform a variety of high-demand military or conventional tasks.

Path planning is one of the most basic technologies of surface unmanned vehicle. Its purpose is to determine the best trajectory of USV navigation by receiving information from sensors under unmanned conditions, and in the actual driving process of USV. According to the error between the actual position and the preset position, continuously modify the travel route affected by external factors to meet the relevant task requirements. The pros and cons of the path planning algorithm not only determine its level of autonomy, but also a prerequisite for the successful realization of the task. In the process of path planning, the main indicators to be investigated are the total path distance and safety. In addition, the quality of the generated trajectory, such as smoothness and continuity, is also within the scope of investigation [2]. Different from robot or unmanned vehicle path planning, due to the particularity of the working environment of surface unmanned vehicle, the USV is required to successfully complete tasks on the surface

W. Fu et al. (Eds.): ICMTEL 2021, LNICST 387, pp. 518–527, 2021.
https://doi.org/10.1007/978-3-030-82562-1_51

with strong interference such as wind, waves and currents, so it is reliable, real-time and anti-interference ability has also become a factor to be considered in design [3].

The path planning of the USV is a type of robot path planning in a broad sense, so many relatively mature robot path planning methods can be considered for use on the unmanned surface vehicle. Currently, the mainstream path planning algorithms can be divided into three categories: path planning methods based on graph search, methods based on sampling, and methods based on intelligent algorithms. Graph search algorithms are usually based on graphics environment modeling. By mapping real-world environmental information to the graph, the path planning problem is transformed into a mathematical problem for solution. Common graph search algorithms include Dijkstra algorithm [4], A* [5] algorithm, and D* [6] algorithm. These algorithms often have complete solutions, as long as there is a feasible route, these methods can be used within a certain time to search for a route that meets the requirements. However, the cost of this completeness is that the search time is closely related to the complexity of the environment, and the time to obtain a feasible path will increase rapidly with the increase of the map scale and the increase of obstacles in the environment. The path planning of intelligent bionics is inspired by animal behavior in nature and can be used to deal with path planning problems in complex environments. Commonly used methods include ant colony algorithm [7], genetic algorithm, particle swarm optimization algorithm, etc. The advantage is that these algorithms have good robustness, and the speed of path planning has been further improved, but at the same time, premature convergence may occur and fall into Local optimal solution, and parameter settings often need to be adjusted adaptively, which is equivalent to bringing greater computational burden to the path planning algorithm. The sampling-based path planning method mainly uses sampling of the state space to find feasible paths on the basis of sampling. The main advantage of this planning method is that it does not need to model the environment and has high applicability. The sampling-based path planning method represents probabilistic road map (PRM) [8] and Rapidly-exploring Random Tree (RRT). Equally, the problem with this method is that the path search tree is generated by random sampling points. The quality of the route largely depends on the quality of the sampling method.

2 Scheme Design

For unmanned surface vehicle, path planning can be divided into two parts: global path planning and local path planning. Global path planning usually first pre-plans an executable route based on environmental information such as electronic charts. On the basis of global planning, USV obtains surrounding environment information through its own sensor detection device, and then uses the local path planning method to achieve obstacle avoidance. The advantage is that the global path is obtained by taking a long time in advance to plan the global path, which avoids the blindness that may occur during the local obstacle avoidance process, and can effectively reduce the time for the local path planning algorithm. The specific process is shown in Fig. 1.

Global path planning usually requires the establishment of an environmental model that contains the necessary information to ensure the navigation of the ship based on the information of obstacles and dangerous areas in the sea in the electronic chart. Then

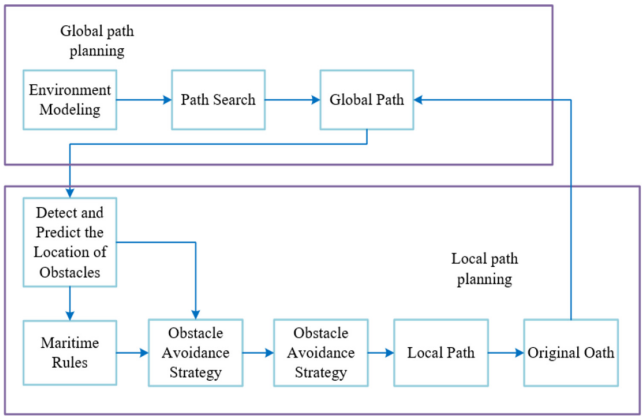

Fig. 1. USV path planning process

find a safe path that can meet various navigation indicators from the established model. Different from global path planning, the local path planning algorithm for USV needs to perceive the information in the surrounding environment in real time based on the multi-type sensors on the vehicle, and dynamically modify the local path according to factors such as distance, wind and waves, to ensure the safe and reliable navigation route.

On the basis of the characteristics of the USV navigation environment, the global path planning method selects the RRT* algorithm. This method is based on incremental search and is particularly suitable for path planning in large spaces. Compared with graph search algorithms like A* or Dijistra, the samples-based RRT* algorithm avoids the limitation that each raster must be searched, and only needs to set the sampling function and step size reasonably according to the size and complexity of the environment, and then global optimal path can be obtained. In the local path planning of the USV, the kinematic parameters must be considered, otherwise the route generated by the planner cannot be actually used on the ship, so the local path planning method chooses the dynamic window approach, which can meet the actual movement of the USV better.

3 RRT*-DWA Algorithm

3.1 RRT Algorithm

It is difficult to solve the problem of path planning when there are nonintegrity constraints or differential constraints. LaValle proposed the RRT algorithm of random sampling in 1998 [10]. He constructed a search tree by using incremental forward sampling, and then searched the tree structure to get the path. The specific process is as follows: after adding the starting position and pose to the random tree, the starting position and pose are directly sampled randomly in the space, the nodes in the whole random tree are traversed, and the nodes nearest to the random position and pose are selected. On the premise of satisfying the constraints, the nodes grow in the direction of the random point with a certain step length until they reach the target region. Starting from the target pose point, the parent node of the current pose point is searched in turn until the starting pose

point is returned to obtain the planning path. The algorithm pseudo-code is shown in Table 1:

Table 1. RRT Algorithm.

Algorithm 1: RRT Algorithm

Input: $\mathcal{M}$,x_{init},x_{goal}
Result: A path Γ from x_{init} to x_{goal}
$\mathcal{T}$.init();
for $i = 1$ *to* n **do**
  $x_{rand} \leftarrow Sample(\mathcal{M})$;
  $x_{near} \leftarrow Near(x_{rand}, \mathcal{T})$;
  $x_{new} \leftarrow Steer(x_{rand}, x_{near}, StepSize)$;
  $E_i \leftarrow Edge(x_{new}, x_{near})$;
  if *CollisionFree*$(\mathcal{M}, E_i)$ **then**
    $\mathcal{T}.addNode(x_{new})$;
    $\mathcal{T}.addEdge(E_i)$;
  if $x_{new} = x_{goal}$ **then**
    Success();

The input of the algorithm includes map M , initial point x_{init} and target point x_{goal}. The purpose is to get a feasible path from the starting point to the end point. The initial point x_{init} is taken as the root node of the random tree, and the *Sample* function is used for random sampling in the space. The point is marked as x_{rand}. Use the *Near* function to traverse the root node of the current random tree and find the closest point x_{near}, and use the *Steer* function to expand a new child node x_{new} on the line of and in a certain step, as shown in Fig. 2(a). If the connected segment does not collide with the obstacle, add the connected segment as a new child node x_{new} and a new edge to the random expansion tree, as shown in Fig. 2(b); if there is a collision, discard it Reselect this point, as shown in Fig. 2(c). Repeat the above process until reaching or near the goal, as shown in Fig. 2(d), a planned path from to is successfully found; if more than a certain number of expansion nodes have not reached the target area or objectively there is no path from the start to the end. If the path is touched, the plan fails.

The superiority of RRT is fast speed and probabilistic completeness, and the whole state space can be searched in a very short time by means of randomly extending child nodes. However, there are still some defects in the basic RRT algorithm, which cannot be directly applied to the path planning of USV, such as the instability caused by the randomness of node expansion. Since the child nodes of the random tree are generated by random sampling in the state space, different paths are planned each time from the same starting point to the end point in the same map. At the same time, the path cost and the path length are not controllable when the child nodes are extended randomly, the path may be far from the optimal path. The trajectory of RRT is generated by random sampling, and the path composed of random extended node sets is usually dithered and contains many unnecessary folding points. The excessive angle of adjacent child nodes

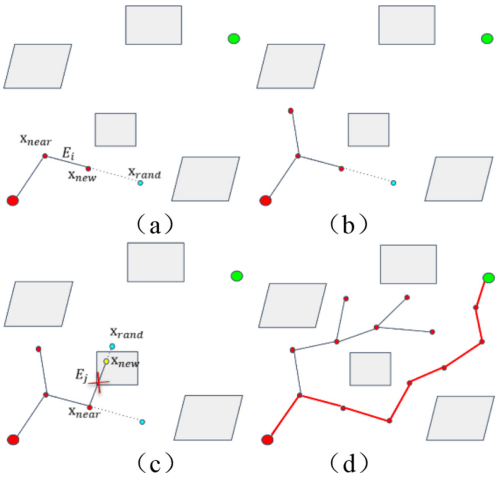

Fig. 2. RRT algorithm example diagram (a) Sampling (b) Extension (c) Collision (d) Results

results in too fast state transition, which cannot be directly applied to the motion planning of USV with non-integrity constraints.

3.2 RRT* Algorithm

Despite the RRT algorithm is a relatively high-efficiency algorithm that can handle path planning problems with non-holonomic constraints, and has great advantages in many aspects, the RRT algorithm does not guarantee that the feasible path obtained is relatively optimized. Therefore, many improvements on the RRT algorithm are also dedicated to solving the problem of path optimization, and the RRT* algorithm is one of them. The main feature of the RRT* algorithm is that it can quickly find the initial path, and then as the sampling points increase, continue to optimize until the target point is found or the maximum number of cycles is reached. The RRT* algorithm is progressively optimized, as the number of iterations increases, the path obtained is more and more optimized, and it is never possible to find the optimal path in a limited time. So in other words, to get a relatively satisfactory optimization path, it takes a certain amount of computing time. Therefore, the convergence time of the RRT* algorithm is a more prominent research problem. But it is undeniable that the cost of the path calculated by the RRT* algorithm is much smaller than that of the RRT.

3.3 Dynamic Window Approach Based on Global Path Planning

The dynamic window approach (DWA) is a local planning method based on velocity sampling [11]. Multiple sets of velocities are sampled in the velocity space, and the trajectory of USV is simulated within a certain period of time. The quality of these tracks are compared according to the preset evaluation function, from which the speed corresponding to the optimal trajectory is selected for execution.

Although the dynamic window approach has good performance in local obstacle avoidance, the path planned in actual operation is prone to lack of purpose. Based on this, this article adds global constraints on the basis of the original DWA, which is to generate RRT* The global route is input into the DWA as a parameter. It can ensure that the route obtained by the planner has good obstacle avoidance performance and will not deviate too much from the original trajectory. According to the limitations and environmental constraints of the USV itself, the sampling space of the speed can be limited within a certain range, as shown in Eqs. (1)–(3).

$$v \in [v_{\min}, v_{\max}], \omega \in [\omega_{\min}, \omega_{\max}] \tag{1}$$

$$v \in [v_c - \dot{v} \cdot dt, v_c + \dot{v} \cdot dt] \tag{2}$$

$$\omega \in [\omega_c - \dot{\omega} \cdot dt, \omega_c + \dot{\omega} \cdot dt] \tag{3}$$

In the formula, v_{min}, v_{max}, ω_{min}, ω_{max} represent the minimum and maximum values of velocity and angular velocity. Respectively, v_c, ω_c represent the current velocity and angular velocity, and $\dot{v}$, $\dot{\omega}$ represent the maximum value of acceleration and angular acceleration.

After sampling multiple sets of velocities, according to the present time window and forward simulation time, the corresponding multiple sets of trajectories are simulated, unsafe trajectories are eliminated from them, and the evaluation functions of other trajectories are calculated. The evaluation function is calculated as formula (4) shown.

$$G = \alpha \cdot hea + \beta \cdot dis + \gamma \cdot vel + \delta \cdot dev \tag{4}$$

Where G is the total evaluation function value. *hea* is used to evaluate the deviation between the current heading and the direction of the target point. *dis* is on behalf of the minimum distance between each point on the current trajectory and obstacles. represented as the current speed, to evaluate the degree of deviation between the current trajectory and the trajectory obtained by the global plan. α, β, γ, δ are the weight coefficients of each index respectively. The physical meaning of the average function formed by the four is: in the local planning, the path avoids obstacles and travels toward the target at a faster speed, while reducing the deviation between the path of the local planning and the global planning.

4 Experiment

4.1 Simulation Conditions

To simulate the actual marine environment, the map size is set to 1000 pixels * 1000 pixels, on which six black areas of different sizes and irregular shapes are distributed, which are regarded as obstacles and dangerous areas. The remaining white parts are regarded as feasible space of USV, shown in Fig. 3. The start point is set at (50,50), and the end point is set at (950,950). In this environment, the RRT, RRT*, and DWA algorithms based on RRT* are compared.

The hardware and software parameters of the simulation experiment are shown in Table 2:

Fig. 3. Simulation of the operating environment of USV

Table 2. Simulation software and hardware parameters.

CPU	AMD r7-4800h
RAM	8.00GB
Storage space	500G
OS	Window 10
Simulation platform	MATLAB 2020a

4.2 Comparison of RRT and RRT* Algorithm

According to the RRT algorithm mentioned above, the simulation was carried out on MATLAB platform, and the RRT algorithm was first run. In practical application, the sampling function is the most effective part of this kind of sampling algorithm. A better route can be obtained only by improving the quality of the sampling points obtained by the sampling function.

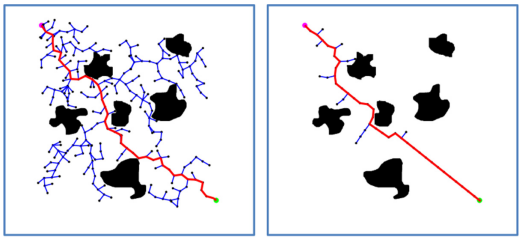

Fig. 4. RRT algorithm results (a) Global random sampling (b) Sample towards end point with a 50% probability (Color figure online)

Simulation of RRT algorithm is shown in Fig. 4. Blue represents the spanning tree and red represents the final path.

It can be seen that when the sampling is added with heuristic functions, the efficiency of the search path has a considerable increase. In Fig. 4 (a), 329 nodes are symbiosis, while in Fig. 4 (b), only 128 nodes are generated to complete the search of the path.Therefore, the design of sampling function is often the core part of RRT algorithm.The sampling function should usually be selected according to the complexity of the environment. For more special environments, such as robot mazes, sampling with heuristic values is less explorative than random sampling for the map.

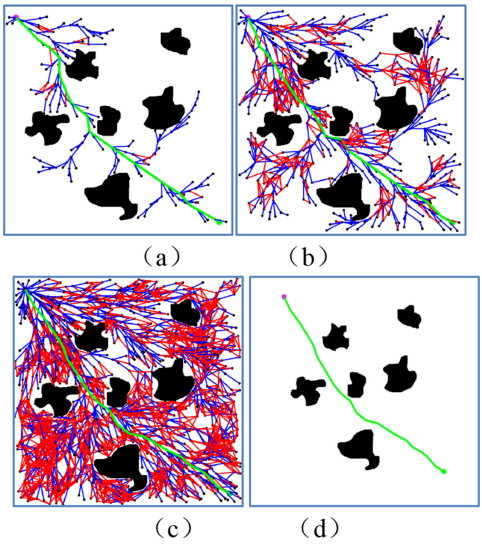

Fig. 5. RRT* algorithm simulation diagram (a) 200 iterations (b) 500 iterations (c) 1000 iterations (d) final route (Color figure online)

The blue line in the figure represents the route when the new node is generated, the red line represents the updated branch, and the green represents the current optimal route. Figure 5 shows the progressive optimality of the RRT* algorithm. As the number of iterations increases, the number of nodes continues to expand, the distance of the planned path is continuously shortened, and the route is smoother. The comparison between the results of the final RRT* generated path and the average results of multiple RRT algorithms is shown in Table 3. It can be seen that although RRT* takes dozens of times as long as RRT, the path length is shortened by 7.6% and the route is more convenient for actual control.

Table 3. Comparison of RRT and RRT* results.

Algorithm	Nodes	Length/pixel	Time/s
RRT	236	1556. 75114	1.05732
RRT*	1500	1437.32537	56.63981

4.3 RRT*-DWA Algorithm

After RRT* has carried out global planning, the global route is shown in Fig. 5(d). The path information is transmitted to the DWA actuator. As USV runs, it receives obstacle information from the sensor. According to the linear velocity and angular velocity at the time, the planner plans the next velocity and corresponding trajectory.

Introduce two dynamic obstacles in the simulation environment, add two black squares on the basis of the original picture and then perform local path planning. The position of USV at some moments is shown in Fig. 6, and the path obtained by the global plan would collide with obstacles, and local planning can successfully avoid collisions.

Figure 6 (a) to (c) reflect the position of USV at a certain moment in actual operation when there are dynamic obstacles nearby. In the final trajectory, Fig. 6(d), it can be seen that due to the presence of two dynamic obstacles, the red actual route of the USV has shifted from the green global route, and the obstacle has been successfully avoided.

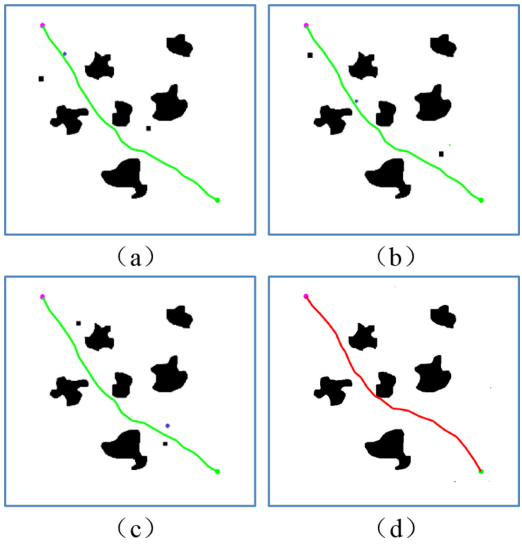

Fig. 6. RRT*-DWA algorithm simulation results (a) t = 20 s (b) t = 40 s (c) t = 60 s (d) final route (Color figure onine)

5 Conclusion

To solve the path planning problem of USV, this paper proposes a dynamic window method based on optimized rapidly-exploring random tree, which is denoted as RRT*-DWA. First, use RRT* to quickly generate an optimal global path, and then DWA algorithm determines the feasible speed sampling space according to the motion parameter constraints of USV, and perform several sets of sampling. According to the evaluation function of the global path, the optimal path is selected to track execution. It is verified

by simulation experiments that the USV can effectively avoid dynamic obstacles, and on the basis of ensuring real-time performance, it can ensure the navigation safety of USV in complex environments.

References

1. Wang, N., Jin, X., Er, M.J.: A multilayer path planner for a USV under complex marine environments. Ocean Eng. **184**, 1–10 (2019)
2. Liu, Y., Bucknall, R.: Path planning algorithm for unmanned surface vehicle formations in a practical maritime environment. Ocean Eng. **97**, 126–144 (2015)
3. Singh, Y., Sharma, S., Sutton, R., et al.: A constrained A* approach towards optimal path planning for an unmanned surface vehicle in a maritime environment containing dynamic obstacles and ocean currents. Ocean Eng. **168**(DEC.1), 187–201 (2018)
4. Rhyd, L.: Algorithms for finding shortest paths in networks with vertex transfer penalties. Algorithms **13**(11), 269 (2020)
5. Deci Edward, L., Ryan Richard, M.: Facilitating optimal motivation and psychological well-being across life's domains. Canadian Psychol./Psychologie canadienne **49**(1), 14 (2008)
6. Stentz, A.: Optimal and efficient path planning for partially-known environments. In: IEEE International Conference on Robotics & Automation. IEEE (2002)
7. Dorigo, M., Caro, G.D., Gambardella, L.M.: Ant algorithms for discrete optimization. Artif. Life **5**(2), 137–172 (1999)
8. Boor, V., Overmars, M.H., Stappen, A.F.V.D.: The Gaussian sampling strategy for probabilistic roadmap planners. In: Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C). IEEE (2002)
9. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Mhs95 Sixth International Symposium on Micro Machine & Human Science. IEEE (2002)
10. Karaman, S.: Sampling-based algorithms for optimal motion planning. Int. J. Robot. Res. **30**(7) 846–894 (2011)
11. Fox, D., Burgard, W., Thrun, S.: The dynamic window approach to collision avoidance. IEEE Robot. Autom. Mag. **4**(1), 23–33 (2002)