Journal of
Marine Science and Engineering

MDPI

Article

A Path-Planning Strategy for Unmanned Surface Vehicles Based on an Adaptive Hybrid Dynamic Stepsize and Target Attractive Force-RRT Algorithm

Zhen Zhang [1], Defeng Wu [1,2,*], Jiadong Gu [1] and Fusheng Li [3]

[1] School of Marine Engineering, Jimei University, Xiamen 361021, China; zhen_zhang2333@outlook.com (Z.Z.); gujiadong2333@outlook.com (J.G.)
[2] Fujian Provincial Key Laboratory of Naval Architecture and Ocean Engineering, Xiamen 361021, China
[3] School of Navigation, Jimei University, Xiamen 361021, China; 201361000041@jmu.edu.cn
* Correspondence: defeng@jmu.edu.cn; Tel.: +86-592-6181728

Received: 8 March 2019; Accepted: 5 May 2019; Published: 8 May 2019

Abstract: It is well known that path planning has always been an important study area for intelligent ships, especially for unmanned surface vehicles (USVs). Therefore, it is necessary to study the path-planning algorithm for USVs. As one of the basic algorithms for USV path planning, the rapidly-exploring random tree (RRT) is popular due to its simple structure, high speed and ease of modification. However, it also has some obvious drawbacks and problems. Designed to perfect defects of the basic RRT and improve the performance of USVs, an enhanced algorithm of path planning is proposed in this study, called the adaptive hybrid dynamic stepsize and target attractive force-RRT(AHDSTAF-RRT). The ability to pass through a narrow area and the forward speed in open areas of USVs are improved by adopting the AHDSTAF-RRT in comparison to the basic RRT algorithm. The improved algorithm is also applied to an actual gulf map for simulation experiments, and the experimental data is collected and organized. Simulation experiments show that the proposed AHDSTAF-RRT in this paper outperforms several existing RRT algorithms, both in terms of path length and calculating speed.

Keywords: path planning; USV; RRT; AHDSTAF-RRT; improved algorithm; dynamic stepsize; target attractive force

1. Introduction

With the increasing frequency of human activities, the rapid consumption of global energy, and the harsh deterioration of environment, the development and application of water surface and marine areas is becoming increasingly extensive [1–3]. Most water operations and tasks need to be accomplished by ships, due to the special working environment of water. At the same time, unmanned equipment is becoming more and more widely used in our society, with the breakthroughs brought about by multiple theories and the development of related technologies, such as artificial intelligence, bionic intelligence, and control science [4]. With the drive of engineering application demand in some bodies of water-such as shoals, lakes, and rivers-there has been an increasing development of unmanned surface vehicles (USVs) in recent decades [5]. The applications of USV include national and civilian uses [6]. Successful applications can be found in diverse areas, such as the replenishment of underway ships, marine exploration, maritime search and rescue, the fishery industry, coastal patrolling, and hydrologic monitoring [7].

The path-planning research of USVs not only determines the level of autonomy of the vehicle, but also influences the reliability of a mission and the likelihood of success [8]. As the algorithm is

the most critical part of USV path planning, this must be elucidated [9]. Optimizing safety, energy consumption, and travelling time are the main aims of a path-planning algorithm.

To address a vehicle's path planning, many kinds of path-planning algorithms have been proposed by researchers [10]. Zeng et al. presented an online dynamic path re-planning system for an autonomous underwater vehicle [11]. Aghababa applied a numerical solution of the nonlinear optimal control problem (NOCP) to determine optimal paths in environments with obstacles [12]. Warren presented an algorithm using artificial potential fields to aid in path planning [13]. Petres et al. presented a novel fast marching (FM)-based approach to address path planning [14]. In the sense of energy saving, the effect of the sea current in path planning was presented and an A* search algorithm with a time-optimal cost was proposed by Garau et al. [15]. Several computational approaches comprising evolutionary methods have been applied in path planning for marine vehicles. Song et al. proposed an algorithm combining nonholonomic constraints of vehicles with a double extended rapidly-exploring random tree (RRT), which not only improves the efficiency of searching but also guarantees the feasibility of the path at the same time [16]. Singh et al. proposed a grid-based path-planning approach for USVs considering static and moving obstacles together with sea surface currents [17]. Du et al. proposed an algorithm combining the environmental constraints and the constraints of intelligent vehicles with RRTs [18].

The advantage of the RRT algorithm is that it can be used to plan a path in a complex environment without building a spatial modeling [19,20]. Meanwhile, the RRT-based path-planning method also has some deficiencies, such as high randomness, inflexible stepsize, slow rate of calculation, and so on. The route generating from path planning is seriously affected by these deficiencies [21]. In the sense of optimizing route generation, improving the speed of calculation, maximizing ease of control, and saving energy, these deficiencies of RRT should be overcome by forming a new algorithm that is a hybrid of superior mechanisms.

Some improved methods have also been used by many studies. Lin et al. proposed an adaptive RRT algorithm based on dynamic stepsize for path planning, in order to solve the problem that the traditional RRT algorithm easily falls into a local minimum area when applied to unmanned aerial vehicles [22]. Melchior et al. defined a new fractional attractive force for robust path planning of mobile robot, and this method obtained robust path planning despite robot mass variations [23]. Liu et al. introduced an algorithm based on RRT by adding a gravity component of the target to solve the disadvantages of the high randomness and long path length [24]. Cao et al. added a dynamic stepsize and target gravity to the basic RRT algorithm. They applied this to the path planning of an agricultural mechanical arm and obtained a good result for litchi picking [25].

However, it should be noted that some shortcomings still exist in these traditional RRT algorithms, such as the high randomness of the search tree growth directions, especially when these path-planning algorithms are applied in certain open areas [26]. In addition, there is often a complex environment with many narrow areas and open areas in the USV's workspace. To deal with these deficiencies of the existing RRT-based algorithms, a novel algorithm is proposed in this paper that is improved through a hybrid of the basic RRT and two mechanisms: dynamic stepsize and target attractive forces. The proposed algorithm is used to improve the performance of USV tasks in a complex workspace. To verify the effect of the enhanced algorithm, this algorithm must be applied to a simulation experiment before being put into a real ship. A real map of a gulf which has a complex terrain was selected as the simulation area, and some necessary processing was done on this map for simulation. In addition, there were some points set for testing according to actual work requirements. Simulation results show that the proposed algorithm definitely optimizes the path-planning algorithm for USVs, especially in a few particular areas.

This paper is organized as follows. In the current section, the necessity of path-planning research has been described with the relevant development and background of USVs. In Section 2, the problem definition is presented and some symbols are defined. In Section 3, the basic RRT path-planning algorithm is given, its merit and demerits are analyzed, and the proposed adaptive hybrid dynamic

stepsize and target attractive force-RRT(AHDSTAF-RRT) is also described in detail. In Section 4, after describing the experimental preparation, the results of the simulation are presented and analyzed. Finally, in Section 5, conclusions and further work are given.

2. Problem Definition

The setup of the path-planning algorithms addressed in this paper is described in this section. The primary task of the path-planning problem is to find a feasible path from the initial state to the goal region in the shortest distance and least amount of computation time possible [19,27].

The optimal path-planning problem is formally composed of a cost function, boundary conditions, and parameters to optimize the cost function. The state space, or configuration space, is represented by the set $X \in \mathbb{R}^n$, $n \in \mathbb{N}$ and $n \geq 2$, where $x \in X$ is a particular configuration of the USV and n is the dimension of state space. X_{obs} is the set of states representing obstacle configurations and are forbidden regions for the USV, and $X_{obs} \subset X$ in general. The set of states X_{free} denotes the traversable states for the USV. Let *Tree* represent the RRT's tree growing from the initial and goal state. Let V and E be the vertices andedges of *Tree* such that $Tree = (V,\ E) \subset X_{free}$. Let $x_{start} \in X_{free}$ and $x_{goal} \in X_{free}$ represent the initial and goal state, respectively.

Definition 1. *Feasible path planning is performed to find a feasible path τ, such that $\tau : [0,1] \to \left\{\tau(0) = x_{start},\ \tau(1) = x_{goal}\right\} \in X_{free}$, let τ be a collision-free path be denoted by $\tau : [0,1] \in X_{free}$, for a given path planning problem $\left(X_{free},\ x_{start}, x_{goal}\right)$.*

Definition 2. *For a given path planning problem $\left(X_{free},\ x_{start}, x_{goal}\right)$, let $c(\tau)$ be the cost to reach x_{goal} along a path τ. The cost function $c(\tau)$ can hence be formally determined by the following:*

$$c(\tau) = \int_0^1 |\tau|, \{\tau : [0,1] | \tau(0) = x_{start}, \tau(1) = x_{goal}, \tau : [0,1] \to X_{free}\} \tag{1}$$

3. Path-Planning Algorithms and Improvements

Path planning is a critical step for the USV, as the level of endurance, efficiency, and autonomy of a USV are affected by the path-planning strategy [28]. In this section, the basic RRT algorithm and two improved RRT algorithms are first presented based on the current path-planning research. Building off the presented algorithms, the adaptive hybrid dynamic stepsize and target attractive force-RRT(AHDSTAF-RRT) is proposed, and the procedures of the AHDSTAF-RRT are also introduced.

3.1. Basic RRT Algorithm

LaValle proposed a concept of a rapidly-exploring random tree (RRT) as a randomized data structure that is designed for a broad class of path-planning problems [29]. Basic RRT is a data structure and algorithm that is designed for efficiently searching non-convex high-dimensional spaces, and it is constructed incrementally in a way that quickly reduces the expected distance of a randomly-chosen point to the tree.

The basic RRT algorithm, presented in Algorithm 1, is a qualified path-planning algorithm. The basic RRT begins with a tree rooted at the only start node x_{start} with no edge and incrementally extends the tree of collision-free paths. When RRT connects a new state x_{new}, it needs to generate other nodes to create its final path. First, a node would generated through goal node or from free space randomly, called x_{sample}; second, the nearest neighbor node of node x_{sample}, x_{near} would be searched out of existing tree; third, another node x_{new} would be searched in a circle with radius equaling one unit stepsize and centered at x_{near}, to find node x_{new} that located in line between node x_{near} and node x_{sample}, and makes a path with the line segment from the node x_{near} to the node x_{new}; then the node x_{new} would be abandoned if the line segment from node x_{near} to node x_{new} is not collision-free; or the node x_{new} will be added to tree if the line segment is in the collision-free space; final, it would restart node x_{sample}

generating and x_{new} searching, until the distance between goal and node in the tree is smaller than given value, which is end of this calculation.

Algorithm 1 Basic Rapidly-Exploring Random Tree (RRT)

RRTmain()
1. $Tree = x_{start}$
2. $x_{new} = x_{start}$
3. **while Distance**(x_{new}, x_{goal}) < ErrTolerance**do**
4. x_{sample} = **SampleTarget**()
5. x_{near} = **NearestVertex**($Tree$, x_{sample})
6. x_{new} = **ExtendTowards**(x_{near}, x_{sample})
7. $Tree$.**add**(x_{new})
8. **end while**
9. **return Trajectory**($Tree$, x_{new})

The pseudocode of SampleTarget(), which is one of functions in the basic RRT algorithm, is presented in Algorithm 2.

Algorithm 2 SampleTarget()

1: **if Rand()<GoalSamplingProb then**
2: **return** x_{goal}
3: **else**
4: **return RandomConfiguration**()
5: **end if**

The algorithms discussed above contain the following functions:

- Distance: Given two states x_{new} and x_{goal}, it returns the distance between these two states by a distance function. Euclidean distance is usually used for the distance function.
- SampleTarget: Given a graph configuration space, it returns a state x_{sample}, which is either the state x_{goal} or a state randomly selected from the configuration space.
- NearestVertex: Given a graph tree and a state x_{sample}, it returns the state x_{near} that is the nearest state to x_{sample} on the graph tree.
- ExtendTowards: Given two states x_{near} and x_{sample}, it returns a state x_{new} that is located on the line segment from x_{near} to x_{sample}, and such that the distance between x_{near} and x_{new} is equal to unit stepsize.

The objective of the basic RRT algorithm is to start from an initial state x_{start} located in the collision-free space and find a path to the goal state x_{goal}. This is done by continuously adding nodes to the *Tree*, which is grown from the starting state x_{start} in the collision-free space until the distance between the goal and a node in the *Tree* is within the visible range.

Time complexity and space complexity describe how much time and space, respectively, are required by the given algorithm. Large O notation is often used to describe the complexity of an algorithm [20,30].

The time complexity analyses of the basic RRT algorithm, as shown in Algorithm 1, are as follows. Time complexity is usually defined as the number of calls for the most time-consuming procedure, which is the procedure of adding the vertex x_{new} in basic RRT. For a problem with a data size of N samples, the time it takes to add vertex x_{new} to the tree can be calculated as the sum of each step:

$$T(N) = T_{sample}(N) + T_{near}(N) + T_{extend}(N) + T_{add}(N), \quad (2)$$

where T_{sample}, T_{near}, T_{extend}, and T_{add} correspond to the functions in lines 4–7 of Algorithm 1, respectively. With data size of N samples, T_{sample}, T_{near}, and T_{add} are all simple operations, and can be completed in linear time; thus, the time complexity of them is $O(N)$. The extension time, T_{extend}, of the tree can take considerably longer if collision checking or other intelligent extension strategies are used. However, these calculations do not depend on the number of vertices already in the tree, and the time complexity for inserting a vertex is $O(N * \log(N))$. Each time the nearest vertex has to be found, the distance from the vertex x_{sample} to all previously added vertices must be calculated. Thus:

$$T_{extend}(N) = O(N * \log(N)). \tag{3}$$

By adding the derived complexity for the sample, near, extend, and add operations, the combined time complexity for the basic RRT algorithm is

$$T(N) = O(N) + O(N * \log(N)) + O(N) + O(N) \approx O(N * \log(N)). \tag{4}$$

Therefore, the time complexity of the basic RRT is $O(N * \log(N))$ for a data size of N samples.

Space complexity is defined as the amount of memory space used by the given algorithm. Basic RRT maintains a tree $Tree = (V, E)$, and the size of the tree determines the amount of memory space. The space complexity of the basic RRT is $|Tree| \in O(N)$, because the size of $Tree$ can be calculated as the sum of the absolute values of V_N and E_N : $|Tree| = |V_N| + |E_N|$. Clearly, $V_N = N$, and $E_N = N - 1$. Therefore, $|Tree| = |V_N| + |E_N| = 2N - 1 \in O(N)$, and the space complexity of the basic RRT is $O(N)$.

3.2. Improved Methods of RRT

While a basic RRT algorithm alone is insufficient to solve a path-planning problem, it is customarily considered as a component that can be incorporated into the development of a variety of different planning algorithms. Therefore, relative to RRT's shortcomings, which are randomness and a weak ability to pass through a narrow area, two mechanisms-dynamic stepsize and forces due to target attraction-are chosen and added to the basic RRT algorithm to solve its deficiencies.

3.2.1. Target Attractive Force-RRT

Due to the high randomness, there are many cases in which the RRT algorithm tree branches grow in an irrelevant direction. To solve this deficiency, the mechanism of a target attractive force was introduced into the basic RRT algorithm, forming the target attractive force-RRT(TAF-RRT) algorithm.

The following is the method of the basic RRT with a target attractive force added:

$$\theta_1 = arctan\Big(\big(x_{sample}[1] - x_{near}[1]\big)/\big(x_{sample}[0] - x_{near}[0]\big)\Big), \tag{5}$$

$$\theta_2 = arctan\Big(\big(x_{goal}[1] - x_{near}[1]\big)/\big(x_{goal}[0] - x_{near}[0]\big)\Big), \tag{6}$$

$$\theta = k_1\theta_1 + k_2\theta_2, \tag{7}$$

where:

- θ_1 is the angle between the line pointing from x_{near} to x_{sample} and the coordinate axis;
- θ_2 is the angle between the line pointing from x_{near} to x_{goal} and the coordinate axis;
- θ is the final angle between the direction of the branch node growth and the coordinate axis;
- k_1 and k_2 represent the coefficients of θ_1 and θ_2, respectively. In general, $k_1 + k_2 = 1$.

Every time a new node x_{new} generated, it would be slightly biased toward the target node x_{goal} from the original direction toward the sample node x_{sample}, with the bias angle θ generated by target attractive force. Due to this mechanism added, the final collision-free path τ and cost of path $c(\tau)$ would change each time the biases were added together after the process of path planning.

Compared to the basic RRT algorithm, branches of the TAF-RRT are more likely to grow in the direction of the target in an open area after adding the target attractive force. Therefore, the randomness and the length of the path created by this method are reduced. The number of branches represents the successful attempts of connected samples to the tree before a feasible path generated by this algorithm.

3.2.2. Dynamic Step Size-RRT

Another problem with the RRT-based algorithm is that it is difficult to pass through narrow areas of a map because the RRT-based algorithm when the stepsize p is too large. Another consequence of this shortcoming is that the branches grow too slowly in an open area with a small stepsize, which results in the lower speed of the algorithm. An approach is proposed that aims to solve this problem by adding the mechanism of a dynamic stepsize.

The principle of this improved algorithm, dynamic stepsize-RRT(DS-RRT), is that the stepsize of this algorithm is variable depending on different situations of the *Tree*, threshold d, and node x_{near}. When branches growing in narrow area, the stepsize p will become smaller, and likewise will become larger when they face an open field:

$$p = \begin{cases} p_1, & Distance(x_{near}, X_{obs}) < d \\ p_2, & Distance(x_{near}, X_{obs}) \geq d \end{cases}, \tag{8}$$

After the dynamic stepsize mechanism is added, the modified algorithm has a better passage ability in narrow area with smaller stepsize, and also has an higher speed in open area with larger stepsize. Usually, with the fixed initial stepsize p_0, the value of p_1 is a half of p_0, and the value of p_2 is 1.0 to 1.2 times p_0, which are generated by trial and error.

3.2.3. Adaptive Hybrid Dynamic Step Size and Target Attractive Force-RRT

To modify both deficiencies of the RRT algorithm mentioned above, the two described mechanisms are simultaneously added to the basic RRT algorithm, forming the dynamic stepsize and target attractive force-RRT (DSTAF-RRT) algorithm.

However, some test experiments show that the modified algorithm DSTAF-RRT sometimes performs unsatisfactorily in segmental processes, especially in narrow areas, and sometimes even worse than the singly-added mechanisms. Through analysis, it was found that the target attractive force usually hinders the correct development of the tree in the narrow area, because the branch nodes easily hit obstacles due to the attractive force impact and thus grow unhealthily. In view of the above, the following proposal was proposed: when RRT branch nodes grow in a narrow area, the dynamic stepsize mechanism should be added into the basic RRT algorithm and the weight of the target attractive force should be reduced or even completely removed. Likewise, when RRT branch nodes grow in a very open area, the target attractive force and dynamic stepsize should be added simultaneously, and the weight of the target attractive force should be increased. Ultimately, the adaptive hybrid dynamic stepsize and target attractive force-RRT (AHDSTAF-RRT), is born.

Algorithm 3: AHDSTAF-RRT(Improved RRT)

Improved RRTmain()

1. $Tree = x_{start}$
2. $x_{new} = x_{start}$
3. **while Distance**(x_{new}, x_{goal}) < ErrTolerance**do**
4. x_{sample} = **Sample Target**()
5. x_{near} = **NearestVertex**($Tree, x_{sample}$)
6. **if CheckObstacle**(x_{near}, X_{obs}, d)
7. **ExtendTowards←AttractiveForce**(x_{goal}, x_{near});
8. p = **DynamicSize**(T, X_{obs});
9. **else**
10. **ExtendTowards←Reduce AttractiveForce**(x_{goal}, x_{near});
11. p = **DynamicSize**(T, X_{obs});
12. **endif**
13. x_{new} = **ExtendTowards**(x_{near}, x_{sample})
14. $Tree$.add(x_{new})
15. **end while**
16. **return Trajectory**($Tree$,x_{new})

The AHDSTAF-RRT algorithm, presented in Algorithm 3, is similar to the basic RRT algorithm in general, except in the critical steps where the adaptive hybrid adjustment is added through a judgement program statement.

Algorithm 3 employs the following functions:

- CheckObstacle: Given the node x_{near}, obstacle X_{obs}, and threshold d, it returns whether the distance between the node x_{near} and obstacle X_{obs} is bigger than the threshold d. This is used toadjustthe values of the stepsize p and the attractive force. Usually the threshold d is 1.5 to 2 times larger than the stepsize p, which is generated by trial and error.
- AttractiveForce: Given two nodes x_{goal} and x_{near}, this puts a minor force on x_{new} when connecting x_{new} to x_{near} with the function ExtendTowards. The addition of a minor bias angle θ_2 in the ExtendTowards function represents the attractive force from the target.
- DynamicSize: Given a graph $Tree$ and X_{obs}, it returns a dynamical value of the stepsize p. It dynamically generates the stepsize p depending on its input status, providing a numerical stepsize value which is relevant to the current situation.

A flow chart summarizing the improved AHDSTAF-RRT algorithm is shown in Figure 1.

The time complexity analysis of the improved RRT algorithm is as follows. The time it takes to add x_{new} vertices to the tree can be calculated as the sum of the time for N samples of each of the lines 4–14 in Algorithm 3:

$$T(N) = T_{sample}(N) + T_{near}(N) + T_{extend}(N) + T_{add}(N) + T_{check}(N). \quad (9)$$

where T_{sample}, T_{near}, T_{extend}, and T_{add} are the same as the basic RRT, respectively, so the time complexity of T_{sample} and T_{near} are $O(N)$, and the time complexity of T_{extend} also is $O(N * \log(N))$. The judgement program statement of the improved RRT algorithm, CheckObstacle, is an additional step compared with the basic RRT algorithm. T_{check} is the time complexity of this judgement program statement. Two branch statements in the judgment program statement are the same time complexity, $O(N)$. By adding the derived complexity for the sample, near, extend, add, and check operations, the combined time complexity for the improved RRT algorithm is calculated as the following:

$$O(N) + O(N) + O(N * log(N)) + O(N) + O(N) \approx O(N * log(N)). \quad (10)$$

Therefore, the time complexity of the improved RRT is $O(N * log(N))$.

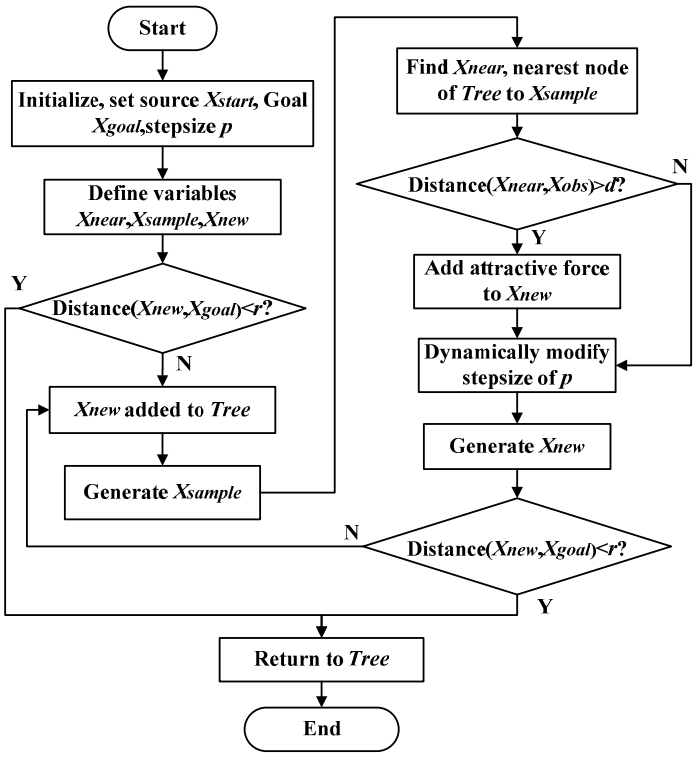


Figure 1. Flow chart of the improved rapidly-exploring random tree (RRT) algorithm.

The space complexity analysis of the improved RRT algorithm is as follows. Similar to the basic RRT, the improved RRT algorithm also maintains a tree $Tree = (V, E)$ and the size of $Tree$ determines the amount of memory space. Given $|Tree| = |V_N| + |E_N|$, obviously the space complexity of the improved RRT is also $O(N)$.

It can be easily seen from the calculation that the improved algorithm keeps the same time complexity and space complexity as the basic RRT, while improving the search speed and keeping the terse advantage of the program statement.

This AHDSTAF-RRT algorithm, improved by adding the adaptive adjustment mechanisms, must be applied in simulations and experiments to test its performance.

4. Simulation Results

It is essential to do an experiment by simulating an actual USV workspace to verify whether the improved AHDSTAF-RRT algorithm is valid. The map area, process, and results of the simulations are included in this section.

4.1. Map Processing and Coordinate System Establishment

The Xinglin Gulf, which is located north of Xiamen, is chosen as the task simulation area due to its complex terrain environment. The map must be processed before it can be used in the simulation experiments. The process of map sharpening is shown in Figure 2. An image of the Xinglin Gulf with a size of 1500 × 1500 pixels is obtained after the map processing.

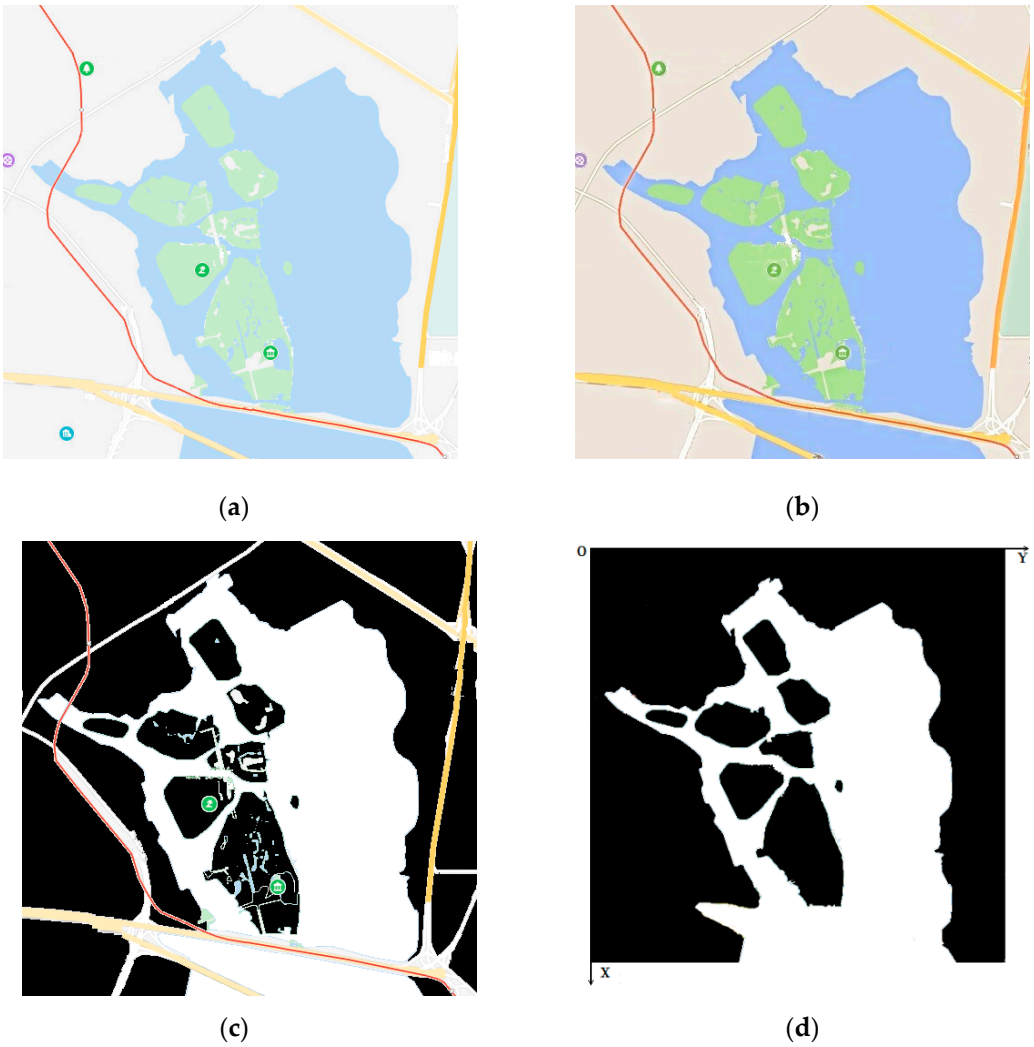


Figure 2. Map processing. (**a**) First step: extracting the image from the map network; (**b**) Second step: sharpening the image. (**c**) Third step: painting the image with white and black color; (**d**) Final step: removing some noisy points from the image.

First, a map of Xinglin Gulf is extracted from the map network and then made into an image with a suitable size, as shown in Figure 2a. After this processing step, information of collision-free space, obstacle space and comparing rule are saved. Second, this image needs to be sharpened to make its outlines clearer and prepare it for the next step, as shown in Figure 2b. Next, special areas of the image need to be painted with a suitable color, usually black and white, as shown in Figure 2c. Usually white represents collision-free space and black represents obstacle space. Finally, some noisy points need to be removed from this image and the image is converted into binary data so that it can be correctly read by a computer, as shown in Figure 2d. It is necessary to explain that this method of map processing only applies to the simulation experiment to verify the effectiveness of the algorithm proposed in this study, while there are other methods to process maps for real tasks.

It is important to establish a coordinate system for map after the map processing and test point selection. The top left vertex of the map is set to the origin of the coordinates, with the x, y axis point from origin toward bottom and right respectively. After processing, this image is transformed into a two-dimensional array space X with obstacle space $X_{obs} = 1$ and collision-free space $X_{free} = 0$, which has a coordinate system and size of 1500×1500 pixels.

4.2. Simulation Experiments of the Algorithm Applied to the Xinglin Gulf Map

The test points are selected according to the technical specifications of actual water sampling. It is well known that sources of water pollution are often in the shore areas of water, therefore all selected test points are located in shallow water areas close to the shore. With the vector space X, the selection of test points refers to an actual water sampling work, and there are six test points and six path segments which appear from these points being connected in turn. The coordinates of Xinglin Gulf test points are listed in Table 1, and the locations of the test points in vector space X of map image are shown in Figure 3.

Table 1. Coordinates of test points.

Point	Coordinates
Point 1	(175,640)
Point 2	(380,560)
Point 3	(540,460)
Point 4	(572,150)
Point 5	(860,440)
Point 6	(1,340,500)

Figure 3. Positions of test points on the map.

From Figure 3, it can be easily seen that the test points are all located in the shore areas of the Xinglin Gulf, with several narrow regions between them. The purpose of this selection is to test whether the performance of the proposed algorithm is improved.

After the test point selection and the determination of the simulation task, five different path-planning algorithms were applied to the simulation of the Xinglin Gulf map in order to test their performance: the basic RRT, DS-RRT, TAF-RRT, DSTAF-RRT, and AHDSTAF-RRT. To avoid the single test randomness, each algorithm will simulate 20 times, and result of each simulation will be recorded to calculate the average quantity.

When these RRT-based algorithms perform path planning, the tree branches extend from the root node x_{start} to search the space of map until the tree stops near the target node x_{goal}. The number of branches represents the successful attempts of connected all random samples to the tree before a feasible path generated by RRT-based algorithms, therefore it could also stand for the computational time cost level. The average lengths of the paths determined from the different algorithms simulated on the Xinglin Gulf map are shown in Table 2, and the average number of branches for each algorithm's growth in the simulation experiments is shown in Table 3.

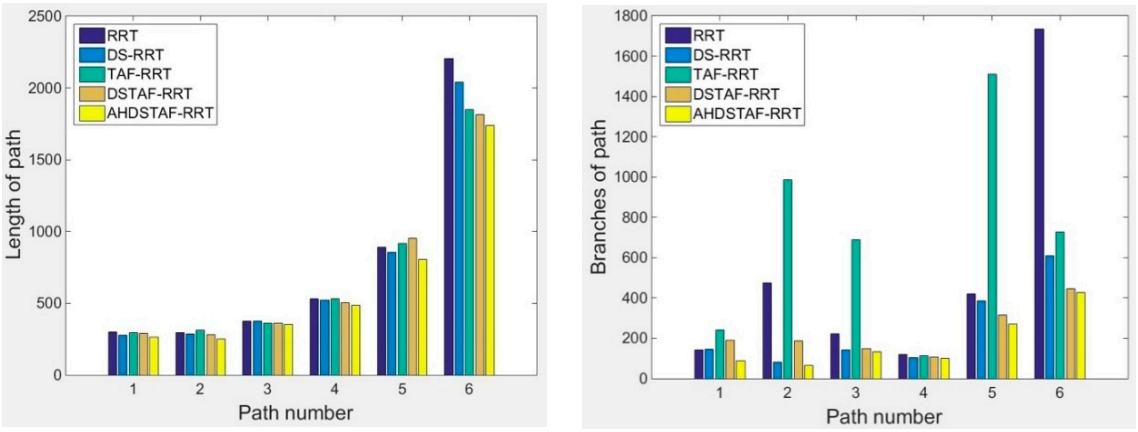
Table 2. Average length of each path through five different algorithms simulated on the Xinglin Gulf map. DS: dynamic stepsize; TAF: target attractive force; AH: adaptive hybrid.

Path Number	1	2	3	4	5	6
Origination	Point 1	Point 2	Point 3	Point 4	Point 5	Point 6
Destination	Point 2	Point 3	Point 4	Point 5	Point 6	Point 1
RRT	297.98114	294.80789	374.09753	529.68422	889.99309	2203.9932
DS-RRT	279.66584	285.57145	373.00868	521.402	856.35898	2039.7009
TAF-RRT	293.88918	314.30826	361.82602	530.41869	917.33472	1848.7661
DSTAF-RRT	292.28904	282.49893	362.41963	503.50753	950.4972	1814.8503
AHDSTAF-RRT	269.72686	252.4269	354.02617	484.70416	807.00211	1737.9915

Table 3. Average number of branches for each algorithm's growth in the simulation experiment.

Path Number	1	2	3	4	5	6
RRT	142.2	475.8	222.3	120.1	422.0	1733.6
DS-RRT	145.3	80.3	141.1	105.5	386.3	610.4
TAF-RRT	242.6	987.1	688.7	112.1	1512.4	727.7
DSTAF-RRT	190.5	186.9	148.6	106.7	316.7	446.5
AHDSTAF-RRT	87.3	66.1	133.7	101.6	270.5	428.8

Histograms comparing the results are shown in Figure 4, the average length of each path through the different algorithms histogram is shown in Figure 4a, and the average number of branches for each algorithm's growth is shown in Figure 4b.



(**a**) (**b**)

Figure 4. Lengths and number of branches for different algorithms simulated on the Xinglin Gulf map. (**a**)Lengths of each path; and (**b**) number of branches of each path.

It can be obviously seen that the basic RRT performed worst in terms of the average length of each path compared with the other improved algorithms. On the other hand, the AHDSTAF-RRT has the best performance among these algorithms, having the lowest average length cost for each test. In terms of the average number of branches, the TAF-RRT algorithm and DSTAF-RRT perform poorer than the basic RRT in narrow areas, which confirms our previous statements. Furthermore, the AHDSTAF-RRT performs among best of these algorithms in terms of the average number of branches.

The final simulation experiment of five algorithms is shown in Figure 5. Each path is composed of six segmented paths, which start point and goal point are respectively set in Table 1 and Figure 3. It is easily seen from Figure 5 that the basic RRT has a high randomness. Additionally, a feasible path in the map of Xinglin Gulf generated by the AHDSTAF-RRT algorithm, which performed better than the other four algorithms both in terms of length and computational rate.

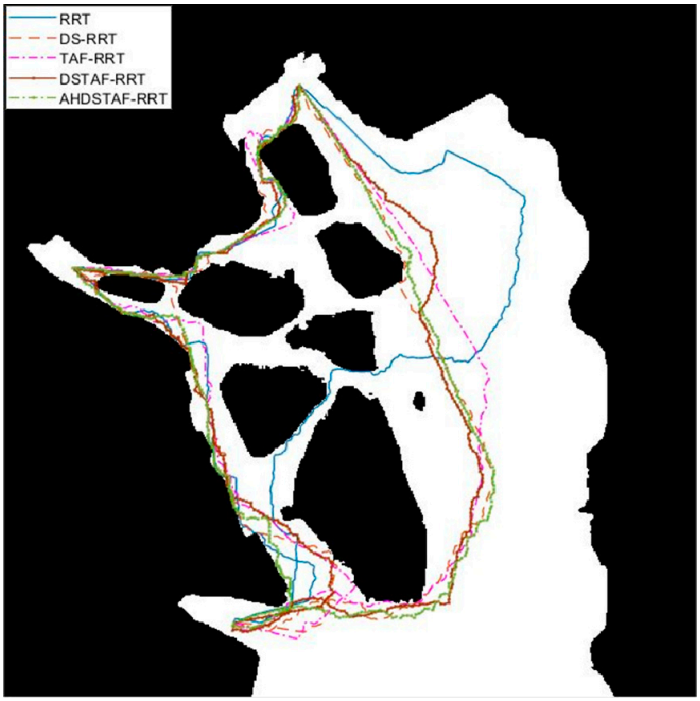


Figure 5. Performance comparison of different algorithms.

5. Conclusions

An RRT-based optimal path-planning algorithm is proposed in this paper, namely the adaptive hybrid dynamic stepsize and target attractive force-RRT(AHDSTAF-RRT). The main idea of this improved algorithm is that it adds two mechanisms—a dynamic stepsize and a target attractive force—into the basic RRT algorithm in the procedure of new node generation and tree growth. This improves on the basic RRT while retaining its beneficial characteristics. Both in terms of length and number of branches, it is obvious that the proposed AHDSTAF-RRT algorithm has created an enhanced path-planning method while keeping the same time and space complexity as the basic RRT. Furthermore, the improved algorithm is not only able to find a better solution to pass narrow areas but is also able to pass open areas with a higher computational speed. In addition, AHDSTAF-RRT is still a tree-extending algorithm, and it can also be combined with any sampling strategy or graph-pruning algorithm to take advantage of any other excellent properties.

In future research, more advantageous mechanisms will be adapted into this algorithm to improve its path-planning performance and computational speed, more complex environments will be studied, and more experiments will be done to observe its passing ability and performance. In addition, with a view to the motion control of USVs, there is a great deal of work to do in the processing of curved sliding after path generation.

Author Contributions: Conceptualization, D.W. and Z.Z.; Methodology, D.W. and Z.Z.; Software, Z.Z. and J.G.; Validation, D.W. and Z.Z.; Formal Analysis, D.W. and Z.Z.; Resources, D.W. and F.L.; Writing-Original Draft Preparation, Z.Z. and D.W.; Writing-Review & Editing, Z.Z., D.W., J.G. and F.L.; Funding Acquisition, D.W.

Funding: This study was financially supported by the National Science Foundation of China (No. 51809113), the Fujian Science and Technology Department (2019H0019), the Fujian Education Department (FBJG20180056, JT180266), and the Program for New Century Excellent Talents in Fujian University (KB16078).

Acknowledgments: We are grateful to Jimei University for the equipment, site, and funding. We also thank Baidu for its free map service.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wu, D.F.; Liu, X.J.; Ren, F.K.; Yin, Z.B. An improved thrust allocation method for marine dynamic positioning system. *Nav. Eng. J.* **2017**, *129*, 89–98.
2. Wu, D.F.; Ren, F.K.; Qiao, L.; Zhang, W.D. Active disturbance rejection controller design for dynamically positioned vessels based on adaptive hybrid biogeography-based optimization and differential evolution. *ISA Trans.* **2018**, *78*, 56–65. [CrossRef]
3. Wu, D.F.; Ren, F.K.; Zhang, W.D. An energy optimal thrust allocation method for the marine dynamic positioning system based on adaptive hybrid artificial bee colony algorithm. *Ocean Eng.* **2016**, *118*, 216–226. [CrossRef]
4. Hanlin, N.; Yu, L.; Savvaris, A.; Tsourdos, A. Efficient path planning algorithms for unmanned surface vehicle. *IFAC-PapersOnLine* **2016**, *49*, 121–126.
5. Mu, D.D.; Wang, G.F.; Fan, Y.S.; Zhao, Y.S. Study on course keeping of POD propulsion unmanned surface vessel. *J. Harbin Eng. Univ.* **2018**, *39*, 274–281. (In Chinese)
6. Ma, Y.; Hu, M.Q.; Yan, X.P. Multi-objective path planning for unmanned surface vehicle with currents effects. *ISA Trans.* **2018**, *75*, 137–156. [CrossRef] [PubMed]
7. Huang, C.F.; Zhang, X.K.; Zhang, G.Q. Improved decentralized finite-time formation control of underactuated USVs via a novel disturbance observer. *Ocean Eng.* **2019**, *174*, 117–124. [CrossRef]
8. Song, R.; Liu, Y.C.; Bucknall, R. A multi-layered fast marching method for unmanned surface vehicle path planning in a time-variant maritime environment. *Ocean Eng.* **2017**, *129*, 301–317. [CrossRef]
9. Peng, Y.; Yang, Y.; Cui, J.X.; Li, X.M.; Pu, H.Y.; Gu, J.; Xie, S.R.; Luo, J. Development of the USV 'JingHai-I' and sea trials in the Southern Yellow Sea. *Ocean Eng.* **2017**, *131*, 186–196. [CrossRef]
10. Niu, H.L.; Lu, Y.; Savvaris, A.; Tsourdos, A. An energy-efficient path planning algorithm for unmanned surface vehicles. *Ocean Eng.* **2018**, *161*, 308–321. [CrossRef]
11. Zeng, Z.; Sammut, K.; Lammas, A.; He, F.P.; Tang, Y.H. Efficient path re-planning for AUVs operating in spatiotemporal currents. *J. Intell. Robot. Syst.* **2015**, *79*, 135–153. [CrossRef]
12. Aghababa, M.P. 3D path planning for underwater vehicles using five evolutionary optimization algorithms avoiding static and energetic obstacles. *Appl. Ocean Res.* **2012**, *38*, 48–62. [CrossRef]
13. Warren, C.W. A technique for autonomous underwater vehicle route planning. *J. Ocean. Eng. IEEE* **1990**, *15*, 199–204. [CrossRef]
14. Petres, C.; Pailhas, Y.; Patron, P.; Petillot, Y.; Evans, J.; Lane, D. Path planning for autonomous underwater vehicles. *IEEE Trans. Robot.* **2007**, *23*, 331–341. [CrossRef]
15. Garau, B.; Alvarez, A.; Oliver, G. Path planning of autonomous underwater vehicles in current fields with complex spatial variability: An A* Approach. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, Barcelona, Spain, 18–22 April 2005; pp. 194–198.
16. Song, J.Z.; Dai, B.; Shan, E.Z.; He, H.G. An improved RRT path planning algorithm. *Acta Electron. Sin.* **2010**, *38*, 225–228. (In Chinese)
17. Singh, Y.; Sharma, S.; Sutton, R.; Hatton, D.; Khan, A. 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.* **2018**, *169*, 187–201. [CrossRef]
18. Du, M.B.; Mei, T.; Chen, J.J.; Zhao, P.; Liang, H.W.; Huang, R.L.; Tao, X. RRT-based motion planning algorithm for intelligent vehicle in complex environments. *Robot* **2015**, *37*, 443–450. (In Chinese)
19. Tahir, Z.; Qureshi, A.H.; Ayaz, Y.; Nawaz, R. Potentially guided bidirectionalized RRT* for fast optimal path planning in cluttered environments. *Robot. Auton. Syst.* **2018**, *108*, 13–27. [CrossRef]
20. Jeong, I.B.; Lee, S.J.; Kim, J.H. Quick-RRT*: Triangular inequality-based implementation of RRT* with improved initial solution and convergence rate. *Exp. Syst. Appl.* **2019**, *123*, 82–90. [CrossRef]
21. Svenstrup, M.; Bak, T.; Andersen, H.J. Minimising computational complexity of the RRT algorithm a practical approach. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 5602–5607.
22. Lin, N.; Zhang, Y.L. An adaptive RRT based on dynamic step for UAVs route planning. In Proceedings of the 5th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 27–29 June 2014; pp. 1111–1114.

23. Melchior, P.; Metoui, B.; Najar, S.; Abdelkrim, M.N.; Oustaloup, A. Robust path planning for mobile robot based on fractional attractive force. In Proceedings of the 2009 American Control Conference, St. Louis, MO, USA, 10–12 June 2009; pp. 1424–1429.
24. Liu, C.J.; Han, J.Q.; An, K. Dynamic path planning based on an improved RRT algorithm for RoboCup robot. *Robot* **2017**, *39*, 8–15. (In Chinese)
25. Cao, X.M.; Zou, X.J.; Jia, C.Y.; Chen, M.Y.; Zeng, Z.Q. RRT-based path planning for an intelligent litchi-picking manipulator. *Comput. Electron. Agric.* **2019**, *156*, 105–118. [CrossRef]
26. Taheri, E.; Ferdowsi, M.H.; Danesh, M. Fuzzy greedy RRT path planning algorithm in a complex configuration space. *Int. J. Control Autom. Syst.* **2018**, *16*, 3026–3035. [CrossRef]
27. Song, R.; Liu, Y.C.; Bucknall, R. Smoothed A * algorithm for practical unmanned surface vehicle path planning. *Appl. Ocean Res.* **2019**, *83*, 9–20. [CrossRef]
28. Blaich, M.; Köhler, S.; Schuster, M.; Schuchhardt, T.; Reuter, J.; Tietz, T. Mission integrated collision avoidance for USVs using laser range finder. In Proceedings of the OCEANS 2015, Genova, Genoa, 18–21 May 2015; pp. 1–6.
29. Kuffner, J.J.; Lavalle, S.M. RRT-connect: An efficient approach to single-query path planning. In Proceedings of the 2000 ICRA Millennium Conference, IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), San Francisco, CA, USA, 24–28 April 2000; Volume 2, pp. 995–1001.
30. An, B.; Kim, J.; Park, F.C. An Adaptive stepsize RRT planning algorithm for Open-chain robots. *IEEE Robot. Autom. Lett.* **2018**, *3*, 312–319. [CrossRef]