



Article

A Cooperative Hunting Method for Multi-USV Based on the A* Algorithm in an Environment with Obstacles

Zhihao Chen ¹, Zhiyao Zhao ^{1,2,3} , Jiping Xu ^{1,2,3}, Xiaoyi Wang ^{1,3,4}, Yang Lu ¹ and Jiabin Yu ^{1,2,3,*} 

¹ School of Artificial Intelligence, Beijing Technology and Business University, Beijing 100048, China; 2130062048@st.btbu.edu.cn (Z.C.); zhaozy@btbu.edu.cn (Z.Z.); xujp@th.btbu.edu.cn (J.X.); wangxy@btbu.edu.cn (X.W.); 2230602063@st.btbu.edu.cn (Y.L.)

² China Food Flavor and Nutrition Health Innovation Center, Beijing Technology and Business University, Beijing 100048, China

³ Laboratory for Intelligent Environmental Protection, Beijing Technology and Business University, Beijing 100048, China

⁴ School of Arts and Sciences, Beijing Institute of Fashion Technology, Beijing 100029, China

* Correspondence: yujiabin@th.btbu.edu.cn

Abstract: A single unmanned surface combatant (USV) has poor mission execution capability, so the cooperation of multiple unmanned surface ships is widely used. Cooperative hunting is an important aspect of multi-USV collaborative research. Therefore, this paper proposed a cooperative hunting method for multi-USV based on the A* algorithm in an environment with obstacles. First, based on the traditional A* algorithm, a path smoothing method based on USV minimum turning radius is proposed. At the same time, the post order traversal recursive algorithm in the binary tree method is used to replace the enumeration algorithm to obtain the optimal path, which improves the efficiency of the A* algorithm. Second, a biomimetic multi-USV swarm collaborative hunting method is proposed. Multiple USV clusters simulate the hunting strategy of lions to pre-form on the target's path, so multiple USV clusters do not require manual formation. During the hunting process, the formation of multiple USV groups is adjusted to limit the movement and turning of the target, thereby reducing the range of activity of the target and improving the effectiveness of the algorithm. To verify the effectiveness of the algorithm, two sets of simulation experiments were conducted. The results show that the algorithm has good performance in path planning and target search.

Keywords: multi-USV swarm; path planning; A* algorithm; target hunting; obstacle avoidance



Citation: Chen, Z.; Zhao, Z.; Xu, J.; Wang, X.; Lu, Y.; Yu, J. A Cooperative Hunting Method for Multi-USV Based on the A* Algorithm in an Environment with Obstacles. *Sensors* **2023**, *23*, 7058. <https://doi.org/10.3390/s23167058>

Received: 26 June 2023

Revised: 7 August 2023

Accepted: 7 August 2023

Published: 9 August 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In the water environment, a single USV has poor ability on task processing and work efficiency. Multi-USV can greatly improve work efficiency, and the multi-USV cooperation has many advantages on exploration ability and task completion ability. Therefore, the research on multi-USV control becomes important [1]. With development of control and communication technology, the multi-USV cooperative hunting has gradually become a hot spot in the field of multi-USV swarm control [2]. The core of the cooperative hunting control is that the multi-USV swarm completes the target hunting task through cooperative control [3]. In recent years, many scholars have studied the multi-USV cooperative hunting control, and many hunting strategies has been proposed. The multi-USV cooperative hunting strategy is generally divided into two parts. First, the path of the USV should be planned. Then, the target is hunted and tracked in real time. In this study, environments with obstacles are defined as water environments with numerous obstacles and constraints. The above constraints include the dynamic features of the target and the smoothness constraints of the planning path.

At present, there are various methods of path planning, which can be divided into global and local path planning based on the cognition of environment information [4].

Global path planning methods can search for an optimal path in a pre-built environment model based on known environment information. Global path planning methods mainly include the A* algorithm [5], Dijkstra algorithm [6], genetic algorithms [7], particle swarm optimization (PSO) [8], and deep reinforcement learning (DRL) algorithms [9]. The above methods usually require long calculation times, and their efficiency is low. Local path planning methods can obtain the surrounding environment's information by using sensors and autonomously plan a collision-free path in partially unknown environments, so they are suitable for dynamic unknown environments. The commonly used methods mainly include the following three categories. The first type is path planning methods based on virtual potential fields, such as the artificial potential field (APF) method [10]. This kind of method is simple in structure and highly efficient. However, due to the obstacles, it has disadvantages of falling into local minima, and even has the potential to not reach the goal. The second type is path planning methods based on sampling, such as the probabilistic roadmap method (PRM) [11] and rapidly exploring random tree (RRT) [12]. These algorithms have fast sampling speed, but the cost of generating paths is higher, the randomness greatly interferes with the planning rate, and sometimes they cannot even find a feasible solution. Thus, these algorithms are often used to solve high-dimensional planning problems with nonholonomic constraints. The third type is intelligent path planning methods based on bionics, such as ant colony optimization (ACO) [13], the artificial bee colony (ABC) algorithm [14], and the imperialist competitive algorithm (ICA) [15]. Such methods are effective in solving complex optimization problems in low-dimensional, and they are easy to find the global optimal solution. However, the above algorithms are easy to fall into a local optimal solution. Meanwhile, the above algorithms have greater randomness during the iterative process and many parameter adjustments are required. These parameters can only be specified through experience or a neural network. Thus, the algorithm efficiency is low.

Among the above algorithms, the A* algorithm is widely used to solve the path planning problem. But the A* algorithm only selects the optimal path according to the length of the planning path without considering the smoothness of the planning path. As a result, the planning path contains many unnecessary turning points, which is suitable to the motion dynamics of USV. To solve this problem, Zhang et al. proposed a new heuristic function combined with the APF, and introduced it into A* algorithm, which effectively reduces the turning points in the planning path and makes the path smoother [16]. Yu et al. improved the expansion mode of the 8-connections of the traditional A* algorithm to the 20-connection, so that the sharpness of turning at the corner can be greatly reduced and the planning path is smoother [17]. Zhang et al. made a minimum circumscribed circle between two adjacent line segments of the turning point, retains the arc of this segment and removes the sharp corners of the segment to make the path smoother [18]. However, the above algorithm does not take into account that the differences in the movement ability of different USV, which leads to the low generality of the above algorithm. The A* algorithm uses the enumeration algorithm to get the optimal path, and the result is low operation efficiency and a long planning time of the A* algorithm. To solve this problem, Liu et al. combined the Voronoi diagram with the grid method to model the map. As a result, the node does not need to be detected too much, and the total amount of planned paths has been reduced; so the planning time has been reduced [19]. Hong et al. adopted the data structure of the minimum heap and 2D array in the A* algorithm, which reduced the time cost of data processing and reduced the planning time [20]. Zhang et al. introduced a 3D bidirectional sector multilayer variable step search strategy into the A* algorithm to reduce the total number of planning paths. At the same time, the efficiency of the algorithm has improved, and the planning time has reduced [21]. The above methods reduce the planning time by changing the search strategy, but they cannot ensure that the distance of the final planning path is the shortest.

Multi USV collaborative hunting is the key to collaborative control of multi USV populations. Currently, many scholars have studied algorithms related to collaborative hunting

with multiple unmanned underwater vehicles. These methods can be divided into two categories: deterministic methods and heuristic methods. The deterministic method uses mathematical tools to solve the virtual capture points and paths of USV. This method has the advantages of low computational complexity, a simple environment, and high efficiency. Representative algorithms include a direct search based on potential field forces [22] and a formation search based on virtual structures [23]. Heuristic hunting by imitating hunting behavior in nature is a method that has a large number of assumptions and can effectively track targets in environments. Representative algorithms include the expulsion ambush method [24] and neural networks based on the hunting method [25]. When the target has strong operability, the expulsion of the AM bus method has high time efficiency, low cost, strong robustness, and fault tolerance [26]. Wang et al. [27] proposed a distributed obstacle avoidance algorithm suitable for multiple mobile robots. This algorithm combines ant colony optimization (ACO) and dynamic window analysis (DWA) to coordinate multiple robot systems through priority strategies, which has high security and global optimality. Guo et al. [28] proposed a distributed collaborative search algorithm and a dynamic target bounding algorithm suitable for quadcopter aircraft clusters, which can effectively search and dynamically monitor dynamic targets in unknown areas. Souza et al. [29] proposed a decentralized multi-agent tracking method using Deep reinforcement learning, and trained a given number of chasers using shared experience. Each agent executes independently at runtime. Although the above algorithms improve the efficiency of target search, they did not achieve preformation before the search. This algorithm has low flexibility. Sun et al. [30] proposed a self-organizing cooperation strategy for multiple unmanned underwater vehicles, dividing the pursuers into a pursuit group and an ambush group based on the escape strategies of evaders under different encirclement states. This method can effectively capture targets and has strong flexibility. Lv et al. [31] assigned tasks based on the multi-layer circular ambush capture model and the characteristics of the pursuit ship. This method can effectively capture targets and has strong flexibility. The above algorithm sets up a pursuit group and an ambush group, which improves the flexibility of the algorithm through collaborative hunting. The above algorithms require high operability for the pursuit group USV, which reduces the practicality of the algorithm.

In summary, there is a problem of unsmooth paths in USV path planning, which is not conducive to USV motion control during sailing. At the same time, most of the existing cooperative hunting algorithms have low efficiency and poor redundancy. Based on the above shortcomings, this paper proposes a cooperative hunting method for multi-USV based on the A* algorithm in an environment with obstacles. First, based on the traditional path planning A* algorithm, a path smoothing method based on the minimum turning radius of USV is proposed. According to the position between obstacles and paths, paths are divided into three scenarios. Second, based on the minimum turning radius of USV, three different methods are used to select new path nodes and then connect these path nodes using ARC. At the same time, the post order traversal recursive algorithm in the Binary tree method is used to replace the enumeration algorithm to obtain the optimal path, which improves the efficiency of the A* algorithm. Finally, in terms of collaborative hunting, a biomimetic multi USV cooperative hunting method was proposed. By simulating the hunting strategy of lion packs, multiple USV packs are preformed along the target's path and ambushed in a U-shaped array. In the collaborative hunting process, the movement of the target in front of the target is limited by the USV, and the USV on both wings of the target limits the direction of the target's turn, thereby limiting the range of target activity.

The main contributions of this work can be summarized as follows:

- (1) A path smoothing method based on USV minimum turning radius was proposed. Based on the A* algorithm, select a new path node and connect it to an arc to make the path smoother. At the same time, the reverse traversal recursive algorithm in the Binary tree method is used to replace the enumeration algorithm to obtain the optimal path, which improves the efficiency of the algorithm and shortens the planning time of the algorithm.

- (2) A biomimetic based multi-USV collaborative hunting method is proposed. The pre-formation of multiple USV groups is conducted independently on the path. Multiple USV groups do not require manual formation. The universality of this algorithm has been improved. In the hunting process, the formation of multiple USV groups is adjusted to limit the movement and rotation of the target, effectively reducing the range of target activities, and improving the effectiveness of the algorithm.

This paper is organized as follows. Section 2 provides the basis content of the algorithm theory involved in this paper. Section 3 provides the modeling of this paper. Section 4 presents the algorithm proposed in this paper. The experiment results are discussed in Section 5, and the paper is concluded finally in Section 6.

2. Preliminaries

2.1. A* Algorithm

The A* algorithm is a classical heuristic search algorithm based on the Dijkstra algorithm combined with the Breadth First Search (BFS) algorithm [32]. The A* algorithm is used to solve the optimal path problem in global path planning. This method continuously approximates the goal by retrieving the nodes, and the optimal path is found. The A* algorithm introduces a heuristic function $h(n)$ to guide the search direction, omitting searches in irrelevant regions, and thus, it has a high search efficiency. The valuation function of the A* algorithm is defined as follows:

$$f(n) = g(n) + h(n) \quad (1)$$

where $f(n)$ is the valuation function from the starting point via node n to the goal point, $g(n)$ is the actual cost function from the starting point to node n in the state space, and $h(n)$ is the heuristic estimation cost function of the optimal path from node n to the goal point. The cost function $h(n)$ is usually expressed in terms of a Euclidean distance as follows:

$$h(n) = \sqrt{(x_n - x_{goal})^2 + (y_n - y_{goal})^2} \quad (2)$$

where (x_n, y_n) is the coordinate of node n and (x_{goal}, y_{goal}) is the coordinate of the goal.

2.2. Binary Tree Method

The binary tree is a tree structure, and the data structure of many problems can be abstracted into the binary tree [33]. The characteristic of the binary tree is that each node can only have two subtrees at most, and the two subtrees can be divided into left and right. The binary tree is a set of n elements which can be composed of empty sets. The set of the binary tree can also be composed of a root and two disjoint subtrees, which can be divided into left and right. It is an ordered binary tree. When the set of binary tree is empty, the binary tree is called an empty binary tree. The element of the binary tree is also called a node. The recursion algorithm of postorder traversal in the binary tree method is a common traversal method of binary tree. The recursion algorithm of postorder traversal in the binary tree method traverses the left subtree at first, then traverses the right subtree, and finally traverses the root. As shown in Figure 1.

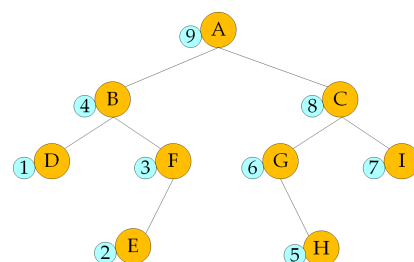


Figure 1. The recursion algorithm of postorder traversal in the binary tree method.

3. Modeling

3.1. USV Modeling

This section provides a mathematical model of the USV, laying the foundation for algorithm research and simulation experiments. The motion coordinate system of the USV on the Earth's surface is shown in Figure 2. Establish a fixed coordinate system on the Earth's surface, where the origin O_e can be any point on the Earth's surface, with the Y_e axis pointing due north and the X_e axis pointing due east, which can be simplified as the $\{e\}$ coordinate system. In practical environments, USVs can heave, sway, surge, yaw, roll, and pitch. This type of USV motion is called a Six degrees of freedom system. To simplify the analysis, this article only considers the yaw, roll, and pitch of the USV. The Six degrees of freedom mathematical model of the USV is simplified to the three degrees of freedom mathematical model of the USV. The three degrees of freedom mathematical model of the USV is represented as:

$$\eta' = J(\eta)v \quad (3)$$

where $\eta = [x, y, \psi]^T$ is the pose vector, which defines the position and heading angle; $v = [u, v, r]^T$ is the speed vector, which includes the forward, sway, and steering speeds; $J(\eta)$ is the transformation matrix from the Earth coordinate system to the USV motion coordinate system [34].

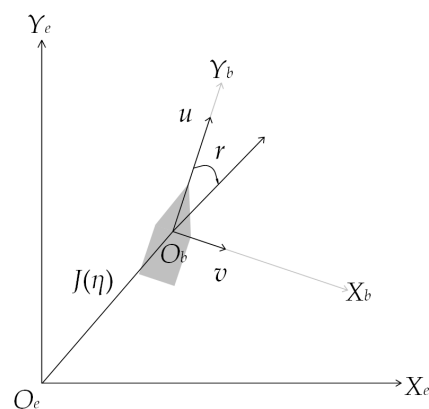


Figure 2. The USV movement coordinate system.

Modeling a real USV using the Unity3D 2020.3.19f1c2 platform to obtain a 3D model of the USV. The actual USV is shown in the figure. The 3D model of Figure 3a and the USV is shown in Figure 3b. The collision module and collision detection module of the Unity3D platform have been added to the 3D model of the USV to facilitate the observation of collisions in the 3D model of the USV. In this paper, the rigid body and kinematics rigid body collider are added to the 3D model of the USV, so that the 3D model of the USV has inertia and gravity. Finally, add the buoyancy module to the 3D model of the USV. The above modeling methods can enable the 3D model of the USV to have real physical characteristics in simulation experiments, and the proposed algorithm can be more effectively validated in simulation experiments.

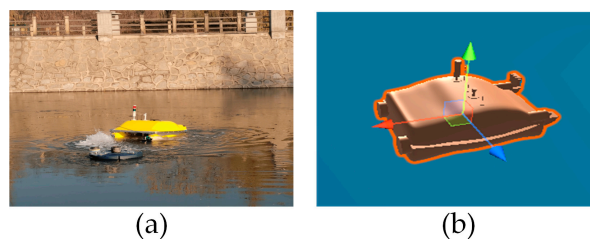


Figure 3. The USV: (a) The Real USV; (b) The 3D USV.

3.2. Obstacle Modeling

This paper uses the Unity3D platform to model obstacles in the map. Because a USV is navigated only on the horizontal plane, the section of an obstacle on the horizontal plane is regarded as an obstacle in the path-planning process. Owing to the uncertainty of obstacles on the water surface, to ensure the navigation safety of a USV, we must leave a safe distance between a USV and obstacles and conduct the expansion modeling for the obstacles. The obstacle coordinate system $\{o\}$ is established with the center of the obstacle as the origin, as shown in Figure 4.

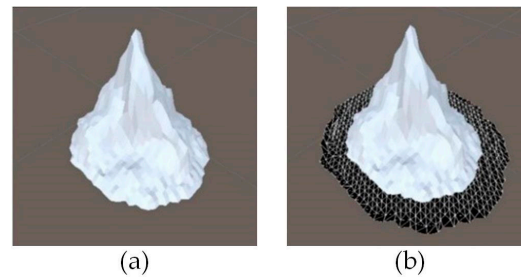


Figure 4. Modeling of obstacle: (a) Static obstacle. (b) Static obstacle after expansion.

The boundary coordinates of obstacles are $(x_{o,i}, y_{o,i})$, $i = 1, 2, \dots, n$. The obstacle is expanded, and the expression is as follows:

$$\begin{cases} \bar{x}_{o,i} = (1 + E) \cdot x_{o,i} \\ \bar{y}_{o,i} = (1 + E) \cdot y_{o,i} \end{cases} \quad (4)$$

where E is the obstacle expansion coefficient.

3.3. Target Modeling

This section is for target modeling, and the motion of the USV on the water surface is shown in Figure 5a. To simplify the analysis, this article imitates the modeling method of USV in Section 3.1 to model the target, considering only the yaw, roll, and pitch of the target, and establishes a mathematical model of a three degrees of freedom target. The target speed is v_{goal} , and the target angular velocity is ω_{goal} . This article uses Unity3D to model the target in 3D, as shown in Figure 5b.

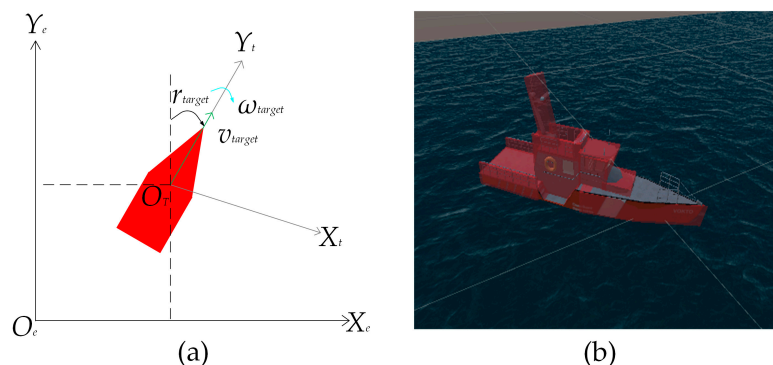


Figure 5. The target: (a) Mathematical modeling of target; (b) 3D modeling of target.

4. Proposed Algorithm

4.1. Improved A* Algorithm

The traditional A* algorithm's planning path does not comply with the motion constraints of USV, and the turning point of the planning path is sharp. Therefore, it is necessary to smooth the planning path of the A* algorithm to ensure that the USV can navigate on the planning path and reach the endpoint.

P is the planning path of the traditional A* algorithm:

$$P = [(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots, (x_i, y_i), \dots, (x_k, y_k)] \quad (5)$$

where (x_i, y_i) is the coordinates of the path nodes in the planning path, and k is the total number of path nodes of the A* algorithm for path planning based on grid graph method.

Let k_c be the curvature of P . r_{max} is the maximum angular velocity of USV. u is the forward speed of the USV. R is the minimum turning radius of the USV. The expression is as follows:

$$R = \frac{|\dot{\eta}'| \cdot (\theta - \varphi)}{2r} \quad (6)$$

where η' and r are obtained according to Formula (3). θ is the angle between the path L' and the X_e -axis. φ is the angle between the path L and X_e -axis.

R_s is the safety range of USV. P is the set of path nodes before smoothing. P' is the set of smoothed path nodes. (x_i, y_i) is the coordinate of the path node where the USV is located, where $i = 1, 2, \dots, k$. (x_t, y_t) is the coordinate of the turning point. t is the abbreviation for *turn*, and (x_t, y_t) is the turning point. (x_n, y_n) is the coordinate of the farthest point of the path section after the turning point, where n is the abbreviation for *node*. L is the length of path from (x_t, y_t) to (x_n, y_n) . L' is the length of path between (x_i, y_i) and (x_t, y_t) . As shown in Figure 6, the red straight line represents the local path before smoothing, and the direction of the red arrow is the forward direction of the path.

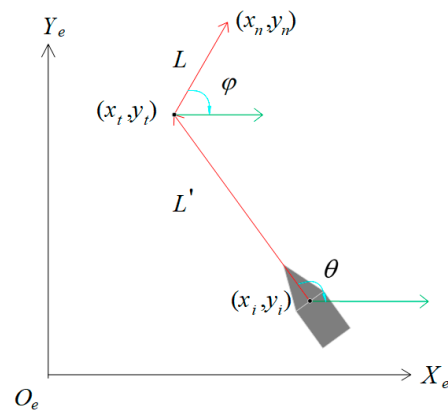


Figure 6. Local path before and after turning point.

When the distance between (x_i, y_i) and the (x_t, y_t) is R_s and the curvature of the path P is greater than the maximum navigable curvature of the USV, i.e., $L = R_s$ and $k_c > r_{max}/u$, start smoothing the path. If the above conditions are not met, it will be considered a smooth planned path and the original path will be retained. The smoothing method is divided into two steps. First, select the path node based on the relative position between the obstacle and the planning path. Second, the path nodes are connected with arcs for smoothing. The relative position between the obstacle and the planning path can be divided into three categories, and different methods are used to select the path node:

(1) When there is no obstacle within the square with the turning point as the center and the side length of $2R_s$, i.e., $\{(x, y) \mid |x - x_t| < R_s, |y - y_t| < R_s\} \cap O = \emptyset$, take the (x_n, y_n) as the $i + 1$ -th path node and save the $i + 1$ -th path node in P' . The coordinates of x_{i+1} and y_{i+1} are as follows:

$$\begin{cases} x_{i+1} = x_i + \frac{L' \cdot \cos \theta + L \cdot \cos \varphi}{\sin(\theta + \varphi)} \\ y_{i+1} = y_i + \frac{L' \cdot \sin \theta + L \cdot \sin \varphi}{\cos(\theta + \varphi)} \end{cases} \quad (7)$$

The result is shown in node 2 on the right in Figure 7.

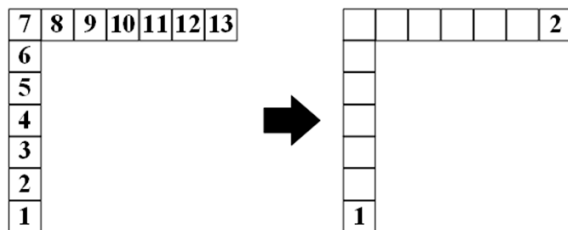


Figure 7. Selection results of the $i + 1$ -th path node when there are no obstacles on both sides of the path.

If there are multiple turning points in the path, the multiple turning points are processed iteratively by (6). In Figure 8, there are two turning points in the planning path. First, re-select the path node according to the first turning point, node 3 on the right side of Figure 8 is obtained by (6) based on node 1. Then, after processing the first turning point, re-select the path node according to the second turning point. Select the point between node 1 and node 3 and away from the second turning point R_s as node 2. Based on node 2, node 4 in the right figure is obtained by (6).

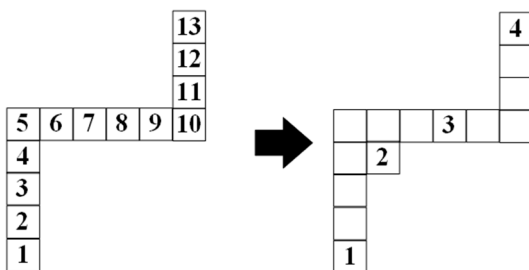


Figure 8. Selection results of path nodes when there are multiple turning points in the path.

(2) When there are obstacles within the square that the midpoint of path L' as the center and the side length of R_s , i.e., $\{(x, y) \mid |x - x_t + (L \cos\varphi)/2| < R_s/2, |y - y_t + (L \sin\varphi)/2| < R_s/2\} \cap O = \emptyset$, set the turning point as the i -th path node. After the i -th path node, take the point inclined by 45 degrees along the path direction and the distance is $\sqrt{2}R$ as the $i + 1$ -th path node, and save the $i + 1$ -th path node in P' . The coordinate values x_{i+1} and y_{i+1} are as follows:

$$\begin{cases} x_{i+1} = x_i + R \cdot \tan(\frac{3}{4}\pi - \theta) \cdot \sin(\frac{3}{4}\pi - \theta) \\ y_{i+1} = y_i + R \cdot \sec(\frac{3}{4}\pi - \theta) \cdot \cos(\frac{3}{4}\pi - \theta) \end{cases} \quad (8)$$

where R is the minimum turning radius of USV. The result is shown in node 3 on the right in Figure 9. Node 2 is the i -th path node selected in the first iteration.

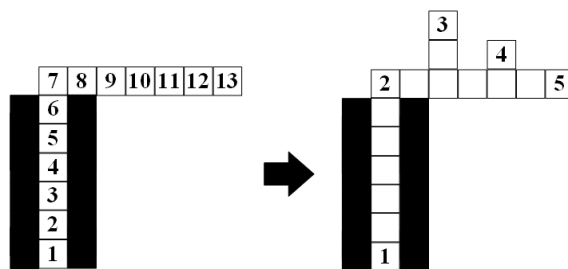


Figure 9. Selection results of path nodes when there are obstacles on both sides of the straight path before the turning point.

After obtaining node 3, set node 3 as the i -th path node for the second iteration. After the i -th path node, take the point inclined by 30 degrees along the forward direction of the path and the distance is R as the $i + 1$ -th path node, and save the $i + 1$ -th path node in P' . The coordinate values x_{i+1} and y_{i+1} are as follows:

$$\begin{cases} x_{i+1} = x_i + R \cdot \tan(\frac{5}{6}\pi - \theta) \cdot \sin(\frac{5}{6}\pi - \theta) \\ y_{i+1} = y_i + R \cdot \sec(\frac{5}{6}\pi - \theta) \cdot \cos(\frac{5}{6}\pi - \theta) \end{cases} \tag{9}$$

The result is shown in node 4 on the right in Figure 9. Then set node 4 as the i -th path node, and repeat the above steps to obtain node 5, and save node 5 in P' .

(3) When there are obstacles within the square that the midpoint of path L as the center and the side length of R_s , i.e., $\{(x, y) \mid |x - x_t + (L \cos \varphi)/2| < R_s/2, |y - y_t + (L \sin \varphi)/2| < R_s/2\} \cap O = \emptyset$. After the i -th path node, take the point inclined by 30 degrees along the negative direction of the path after the turning point and the distance is R as the $i + 1$ -th path node, and save the $i + 1$ -th path node in P' . The coordinate values x_{i+1} and y_{i+1} are as follows:

$$\begin{cases} x_{i+1} = x_i + R \cdot \sec(\frac{5}{6}\pi - \theta) \cdot \cos(\frac{5}{6}\pi - \theta) \\ y_{i+1} = y_i + R \cdot \tan(\frac{5}{6}\pi - \theta) \cdot \sin(\frac{5}{6}\pi - \theta) \end{cases} \tag{10}$$

The result is shown in node 2 on the right in Figure 10. Then, set node 2 as the i -th path node, and repeat the above steps to obtain node 3, and save the node 3 in P' .

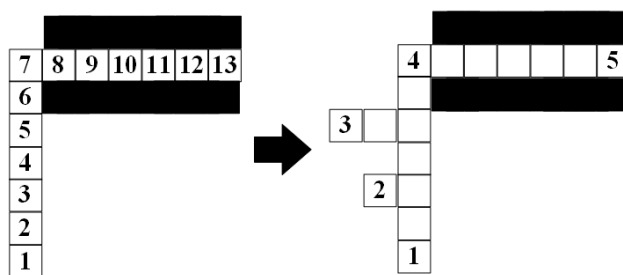


Figure 10. Selection results of path nodes with obstacles on both sides of the straight path after the turning point.

Finally, set node 3 as the i -th path node. Use (7) to obtain node 4, and save the node 4 in P' .

After all the path nodes are obtained, the path nodes need to be connected with arcs for smoothing. Take the path node as the tangent point and the path direction of the path node as the slope to make a straight line:

$$y = Ax + B \tag{11}$$

where A and B are the coefficients of the straight line. This straight line is the tangent of the planning path, which determines the values of A and B and can be obtained through differentiation and algebra. In this study, the Unity3D platform can directly obtain tangent information.

Set (x_{i+1}, y_{i+1}) are the coordinates of the $i + 1$ -th path node. After obtaining the $i + 1$ -th path node, connect the path between (x_i, y_i) and (x_{i+1}, y_{i+1}) using an arc in the circular with radius R_0 , as shown in Figure 11.

In Figure 11, the grid represents the path node, and the connection process of the arc path between node 1 and node 4 is shown from (a) to (d). The red line represents the planning path before smoothing. The blue straight line is obtained by (10). The black curve is a smooth path, which is composed of multiple arcs. The expression of the arc is as follows:

$$\begin{cases} (x - x_{c,i})^2 + (y - y_{c,i})^2 = R_o^2 \\ |A \cdot x_{c,i} - y_{c,i} + B| = R_o \end{cases}, R_o \geq R \tag{12}$$

where $x \in [x_i, x_{i+1}]$, $y \in [y_i, y_{i+1}]$, $(x_{c,i}, y_{c,i})$ is the center coordinate of the arc path from path node i to path node $i + 1$. In Figure 11, $(x_{c,1}, y_{c,1})$ is the center coordinate of the arc path between path nodes 1 to 2, $(x_{c,2}, y_{c,2})$ is the center coordinate of the arc path between path nodes 2 to 3, and $(x_{c,3}, y_{c,3})$ is the center coordinate of the arc path between path nodes 3 to 4. After completing path P' , replace P with P' . R_o is the radius of the circle used for smoothing the path. When the coordinates of two path nodes and the tangent equation of the circle are known, the value of R_o can be obtained through Formula (12).

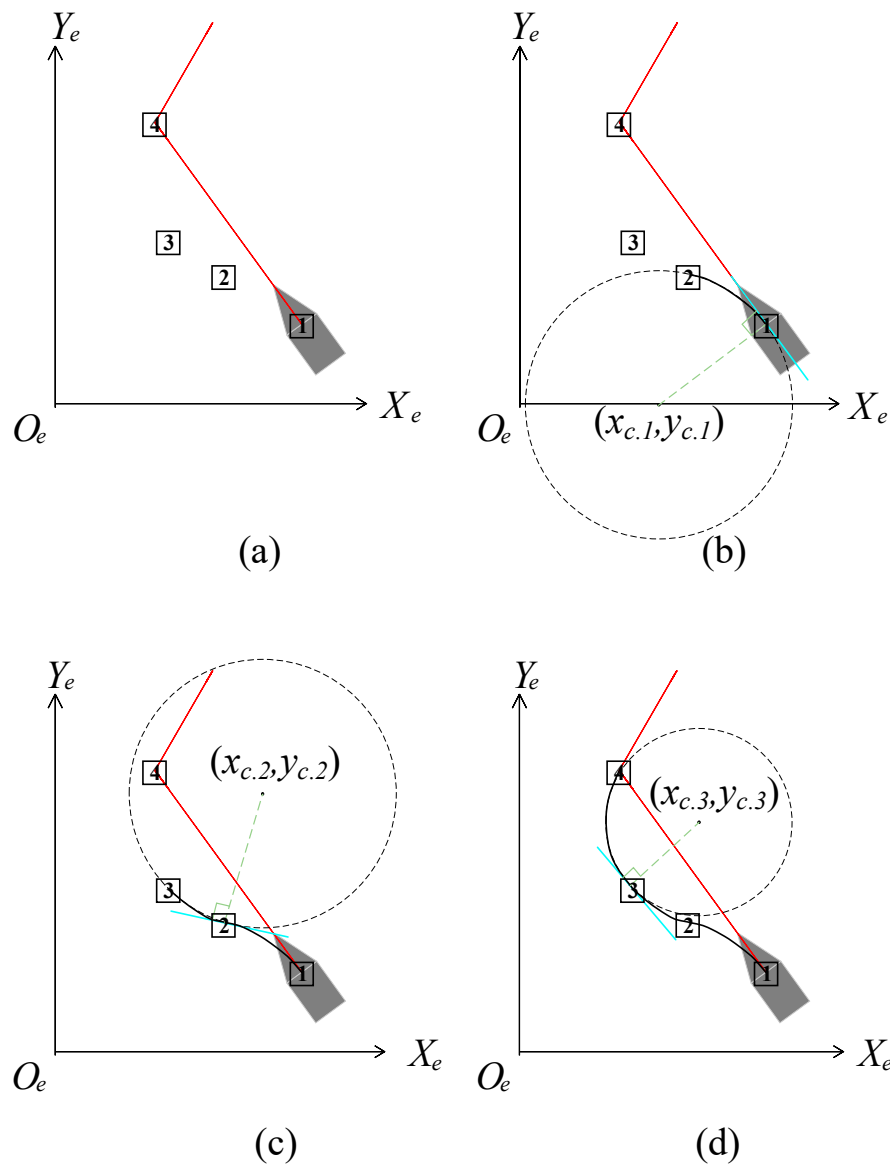


Figure 11. Results of the completing method of the planned path between path nodes: (a) before completing; (b) connection of nodes 1 to 2; (c) connection of nodes 2 to 3; (d) connection of nodes 3 to 4.

Based on the above content, the planning path of the traditional A* algorithm is smoothed. Using Algorithm 1 for path planning between the starting point and the target point, and the steps of the Algorithm 1 are presented by pseudocode:

Algorithm 1: Smoothing method of path turning point

Step1:	Initialize parameters;
Step2:	If $L' = R_s$ and $k_c > r_{max}/u$
Step3:	If $\{(x, y) \mid x - x_t < R_s, y - y_t < R_s\} \cap O = \emptyset$
Step4:	Obtain the path node 2 using (6);
Step5:	If $\{(x, y) \mid x - x_t + (L' \cos \varphi)/2 < R_s/2, y - y_t + (L' \sin \varphi)/2 < R_s/2\} \cap O = \emptyset$
Step6:	Set (x_t, y_t) as path node 2;
Step7:	Obtain the path node 3 using (7);
Step8:	Obtain the path node 4 using (8);
Step9:	Obtain the path node 5 using (8);
Step10:	If $\{(x, y) \mid x - x_t + (L \cos \varphi)/2 < R_s/2, y - y_t + (L \sin \varphi)/2 < R_s/2\} \cap O = \emptyset$
Step11:	Obtain the path node 2 using (9);
Step12:	Obtain the path node 3 using (9);
Step13:	Obtain the path node 4 using (7);
Step14:	Save all path node in P' ;
Step15:	Complete the path of P' using (11);
Step16:	If $k_c > r_{max}/u$
Step17:	Repeat Step3–16;
Step18:	Else Replace P with P' ;
Step19:	Else Replace P with P' .

The traditional A* algorithm uses enumeration algorithms to obtain the optimal path. If all possible situations of a certain type of thing are examined one by one and a universal conclusion is drawn, then this conclusion is reliable. This method is called an enumeration method. The enumeration method utilizes the characteristics of fast computing speed and the high accuracy of computers to test all possible scenarios to solve problems and find answers that meet the requirements. Therefore, the characteristic of enumeration is sacrificing time for a comprehensive answer. In the traditional A* algorithm, although the enumeration method can obtain the optimal path by comparing the lengths of all paths, it does a lot of useless work in obtaining the path, resulting in low efficiency of the algorithm. Therefore, it is necessary to solve the problems of low computational efficiency and long planning time in traditional A* algorithms.

In order to obtain the optimal path from all paths, the algorithm uses the post order traversal recursive algorithm in the binary tree method in reference [33] to replace the enumeration method to obtain the optimal path.

4.2. A Biomimetic Multi USV Swarm Collaborative Hunting Method

The speed of the target is v_{goal} . The location of the target is $P_g(x_{goal}, y_{goal})$. The velocity of the i -th USV in a multi USV group is $v_{ship,i}(x_{ship,i}, y_{ship,i})$. For n virtual structural points, there are n USVs in a multi-USV swarm. The target path is used to obtain the initial virtual structure point P_i of the USV on the path. The coordinates of P_i are $(x_{virtual,i}, y_{virtual,i})$ and satisfy the following formula:

$$\frac{(x_{virtual,i} - x_{goal})^2}{(y_{virtual,i} - y_{ship,i})^2} + \frac{(y_{virtual,i} - y_{goal})^2}{(x_{virtual,i} - x_{ship,i})^2} = \frac{v_{ship,i}}{v_{goal}} \quad (13)$$

where $i = 1, 2, \dots, n$. After obtaining the initial virtual structure points of multi-USV swarm, adjust the position of the virtual structure points on the target path to obtain a U-shaped array. The average value of all initial virtual structural points is $P_a(x_{average}, y_{average})$, which is expressed as follows:

$$\begin{cases} x_{average} = \sum_{i=1}^n \frac{x_{virtual,i}}{n} \\ y_{average} = \sum_{i=1}^n \frac{y_{virtual,i}}{n} \end{cases} \quad (14)$$

P_a is the midpoint P_{mid} of the U-shaped array. Y_{mid} is the ray from P_{mid} to P_g . X_{mid} is a ray that rotates 90 degrees clockwise based on Y_{mid} . The coordinate system is established with P_g as the origin, Y_{mid} as the y -axis, and X_{mid} as the x -axis. The U-shaped array is shown in Figure 12. The expression for a U-shaped array is as follows:

$$y = ax^2 - b \quad (15)$$

where a is the width coefficient, and the larger the value, the wider the U-shaped array, with $a > 0$. b is the distance coefficient, and the larger the value, the farther the U-shaped array is from the target, $b > (4a^2\lambda^2 + 1)/4a$. λ is the minimum distance between the U-shaped array and the target.

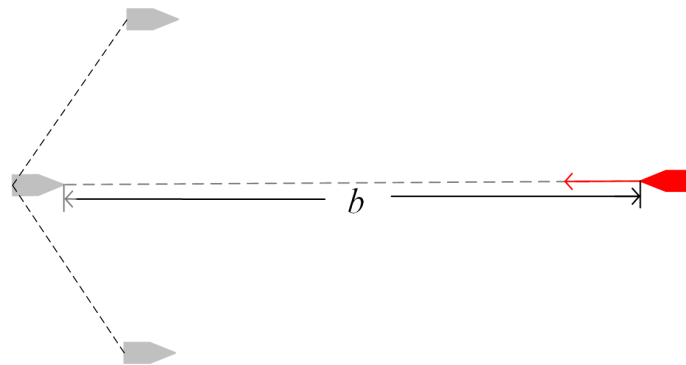


Figure 12. U-shaped array.

The U-shaped array is designed to ambush targets. When the target approaches the U-shaped array, multi-USVs can more conveniently hunt the target.

$x_{mid.1}$ is $x_{average}$, $y_{mid.1}$ is $y_{average}$. $P_{mid.i}$ is arranged on the U-shaped array, and its x -axis coordinate value $x_{mid.i}$ is as follows:

$$x_{mid.i} = x_{mid.i-1} + \frac{v_{ship} \cdot \left(|x_{average} - x_{goal}| + |y_{average} - y_{goal}| \right)}{\eta \cdot v_{goal}} \quad (16)$$

The y -axis coordinate value $y_{mid.i}$ is as follows:

$$y_{mid.i} = \frac{(x_{mid.i})^2}{a} \quad (17)$$

where η is a compaction parameter of the formation, and the larger the value, the more compact the U-shaped array. On the contrary, the smaller the value, the looser the U-shaped array. $i = 2, \dots, n$.

When all USVs reach the formation position, i.e., the distance between all USVs and their corresponding virtual structural points is 0, multi-USV groups begin to pursue the target. The angular velocity of the target is ω_{goal} , the angular velocity of the i -th USV is $\omega_{ship.i}$. The target is the center of the circle, with a radius of R_v . The expression for radius R_v is as follows:

$$R_v = \frac{n \cdot \left(v_{goal} \cdot \sum_{i=1}^n \omega_{ship.i} + \omega_{goal} \cdot \sum_{i=1}^n v_{ship.i} \right)}{\sum_{i=1}^n \left(\omega_{ship.i} \cdot v_{ship.i} \right)} \quad (18)$$

The expression of the circle is as follows:

$$(x - x_{goal})^2 + (y - y_{goal})^2 = R_v^2 \quad (19)$$

$x_{circle.1}$ is x_{goal} , the expression of $y_{mid.1}$ is as follows:

$$y_{circle.1} = y_{goal} - \sqrt{1 - R_v^2} \quad (20)$$

The virtual structure points on the circle are arranged. The expression of the virtual structure points $(x_{circle.i}, y_{circle.i})$ is as follows:

$$\begin{aligned} x_{circle.i} &= \begin{cases} x_{circle.(i-1)} + \frac{4R_v}{n}, 1 < i \leq \frac{n}{2} \\ x_{circle.(i-1)} - \frac{4R_v}{n} - R_v, \frac{n}{2} < i \leq n \end{cases} \\ y_{circle.i} &= \begin{cases} y_{goal} - |x_{circle.i} + 1 - x_{goal}|, 1 < i \leq \frac{n}{2} \\ y_{goal} + |x_{circle.i} + 1 - x_{goal}|, \frac{n}{2} < i \leq n \end{cases} \end{aligned} \quad (21)$$

The improved A-star algorithm was used to obtain the planned path from the current position of the USV to the virtual structure point. Then, form a circle around the target, as shown in Figure 13. In Figure 13, the USV directly in front of the target is the intercepting vessel. The blocking ship is used to block the forward escape route of the target. The USVs on both sides of the target are mainly used to limit the target's turning and be ready to block the target's escape route at any time.

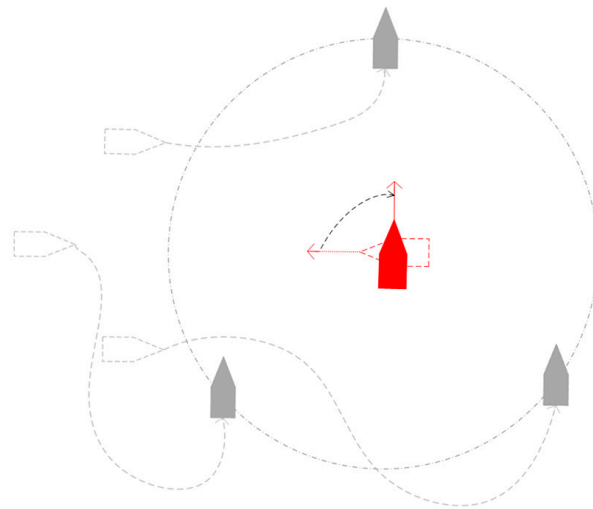


Figure 13. Narrow the surrounding circle for hunting.

Narrowing the surrounding circles, multi-USV swarms have achieved hunting. The target is in the center, resulting in a circle with a radius of R_c . The expression for radius R_c is as follows:

$$R_c = \frac{n \cdot \left(v_{goal} \cdot \sum_{i=1}^n \omega_{ship.i} + \omega_{goal} \cdot \sum_{i=1}^n v_{ship.i} \right)}{\sum_{i=1}^n \left(\omega_{ship.i} \cdot v_{ship.i} \right)} \quad (22)$$

The expression of the circle is as follows:

$$(x - x_{goal})^2 + (y - y_{goal})^2 = R_c^2 \quad (23)$$

$x_{catch.1}$ is x_{goal} , the expression of $y_{catch.1}$ is as follows:

$$y_{catch.1} = y_{goal} - \sqrt{1 - R_c^2} \quad (24)$$

The virtual structural points are evenly arranged in a circle. The position of virtual structural points $(x_{catch.i}, y_{catch.i})$ is obtained by (20). The improved A-star algorithm was used to obtain the planned path from the current position of the USV to the virtual structure point. Multiple USV groups begin to form a capture circle.

When the target attempts to escape the capture circle, adjust the position of the USV to limit the target and make the actual distance d between the USV and the target close to R_c . The distance d between $P_{ship.i}$ and P_g is as follows:

$$d = |P_{ship.i}P_g| = |x_{ship.i} - x_{goal}| + |y_{ship.i} - y_{goal}| \quad (25)$$

where $x_{ship.i}$ is the x -axis coordinate of the i -th USV. $y_{ship.i}$ is the y -axis coordinate of the i -th USV. The USV gathers towards the target direction, reducing the accommodation space between USVs. The escape direction angle of the target is θ_{run} , the angle between the line from the i -th USV to the target and the target direction is θ_i . When the target escapes, adjust the formation of multiple USV groups to block the target's escape route. The virtual structure point $(x_{block.i}, y_{block.i})$ of the i -th USV needs to satisfy the following expression:

$$\begin{cases} y_{block.i} = \tan \theta_i \cdot x_{block.i} - \tan \theta_i \cdot x_{goal} + y_{goal} \\ (x_{block.i} - x_{goal})^2 + (y_{block.i} - y_{goal})^2 = R_c^2 \\ \theta_i = \theta_{run} + (-1)^i \cdot Dimx[\frac{i+1}{2}] \cdot \frac{\pi}{12} \end{cases} \quad (26)$$

where $i = 1, 2, \dots, n$. Finally, the improved A-star algorithm is used to get the planning path from the current position of the USV to the virtual structure point as shown in Figure 14.

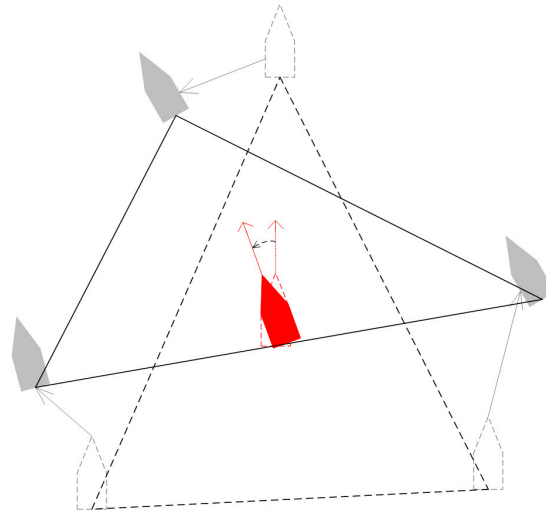


Figure 14. Multi-USV swarm intercepts target.

5. Simulation Experiment and Discussion

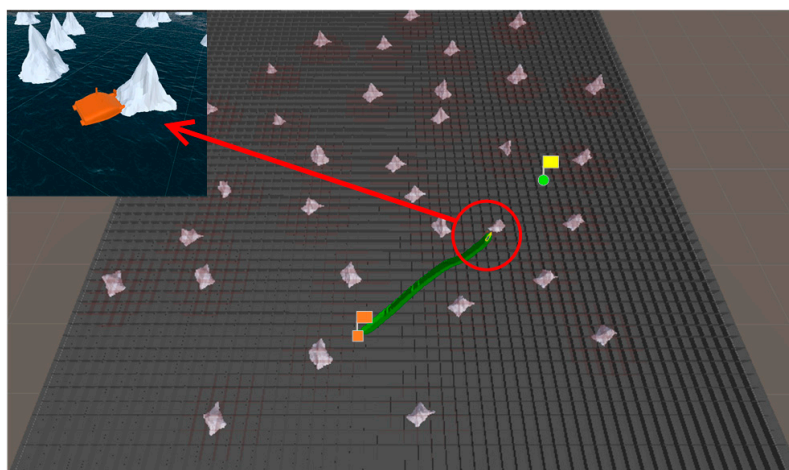
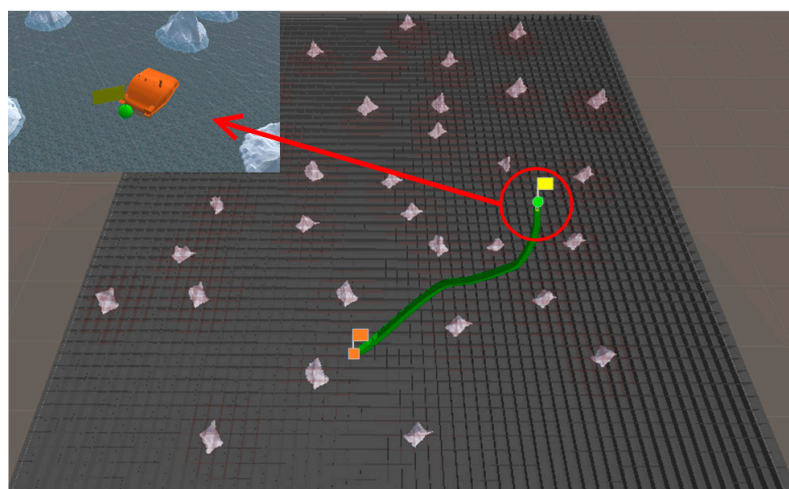
This part is mainly about the comparison and analysis of the experimental results. The simulation experiment used Windows 10 as the operating system and Unity3D as the simulation tool. The hardware platform was an Intel Core i5-10200h processor (Intel Corporation, City of Santa Clara, CA, USA) with a main frequency of 2.4 GHz and 16 GB memory. This experiment is implemented in two steps. First, conduct path planning simulation experiments and compare the experimental results. Second, a comparative experiment between the proposed algorithm and the approaching method is implemented to verify the effectiveness of the proposed algorithm. The relevant parameters of the proposed algorithm and simulation experiment in this paper are shown in Table 1, and their values are obtained from reference [8,11].

Table 1. The algorithm and simulation experimental parameters.

Parameters	Definition	Numerical Value
u (m/s)	Forward speed of USV	15
v_{goal} (m/s)	Forward speed of target	10
r (rad/s)	Angular velocity of USV	1
ω_{goal} (rad/s)	Angular velocity of target	0.7
E	Obstacle expansion coefficient	1.5
a	Width factor	20
b	Distance factor	190
R (m)	Minimum turning radius of USV	3
R_s (m)	Safety range of USV	6

5.1. Simulation Experiment of Path Planning in Scene 1

In Scene 1, the coordinate of the start point is $(-55, -284)$, and the coordinate of the end point is $(291, -49)$. Based on the above map parameters, a comparative experiment between the traditional A* algorithm and the proposed algorithm is conducted to verify the obstacle avoidance ability of the proposed algorithm. The experimental results are shown in Figures 15 and 16.

**Figure 15.** Results of traditional A* algorithm in Scene 1.**Figure 16.** Results of proposed algorithm in Scene 1.

In Scene 1, the traditional A* algorithm does not consider the path smoothing problem. There are turning points in the planning path that do not account for the motion ability of the USV. As a result, USV deviates from the planning path and collides with obstacles, as shown in Figure 15. The proposed algorithm smooths the planning path based on the minimum turning radius of the USV, so that the USV can navigate on the planning path and reach the end point, as shown in Figure 16. Therefore, the proposed algorithm has stronger obstacle avoidance ability than the traditional A* algorithm, and the planning path of the proposed algorithm is smoother than the planning path of the traditional A* algorithm.

5.2. Simulation Experiment of Path Planning in Scene 2

In Scene 2, the coordinate of the start point is (181, -419) and the coordinate of the end point is (-84, 103). The planning path of the traditional A* algorithm is not smooth, the USV deviates from the path and collides with obstacles when tracking the planning path of the traditional A* algorithm. Therefore, in scene 2, based on the above map parameters, the A* algorithm combined with the B-spline curve is selected as the comparison algorithm. The comparison experiment is conducted with the A* algorithm combined with the B-spline curve and the proposed algorithm to verify the effectiveness of the planning path of proposed algorithm and the high operation efficiency of proposed algorithm. The experimental results are shown in Figures 17 and 18, and Table 2.

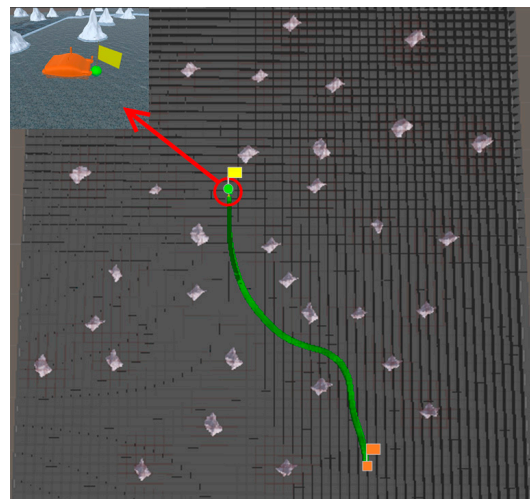


Figure 17. Results of A* algorithm combined with B-spline curve in Scene 2.

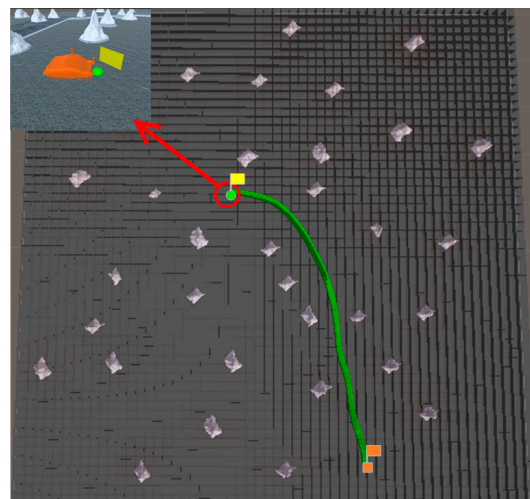


Figure 18. Results of proposed algorithm in Scene 2.

Table 2. Simulation results in Scene 2.

Algorithm Name	Planning Time (s)	Path Length (m)
A* algorithm combined with B-spline curve	0.90211	701.87774
Proposed Algorithm	0.66402	678.60917

Combined with the experimental results in Table 2. It can be seen from Figure 17 that the planning time of the A* algorithm combined with the B-spline curve is longer than the planning time of the proposed algorithm, and the length of the planning path in the A* algorithm combined with the B-spline curve is longer than the length of the planning path in the proposed algorithm. And the total turning angle of the A* algorithm combined with the B-spline curve is also larger than the total turning angle of the proposed algorithm. As can be seen from Figure 18, since the algorithm in this paper smooths the path based on the minimum turning radius of the USV, which shortens the length of the planning path in proposed algorithm and reduces the total turning angle in proposed algorithm, the planning path of proposed algorithm is suitable for the motion ability of the USV. In this paper, the recursion algorithm of postorder traversal in the binary tree method is used instead of the enumeration algorithm to get the optimal path, which improves the operation efficiency of the proposed algorithm and shortens the planning time of the proposed algorithm.

In order to avoid the contingency of the experiment results, three groups of different start points and end points are selected in Scene 2 for the simulation experiments. The result is shown in Table 3.

Table 3. Simulation results of different start and end points under Scene 2.

Start	End	Algorithm	Planning Time (s)	Path Length (m)	Total Turning Angle
(181, −419)	(−84, 103)	A* algorithm combined with B-spline curve	0.90211	701.87774	161°1548′
		Proposed Algorithm	0.66402	678.60917	87°0683′
(−94, −27)	(347, −451)	A* algorithm combined with B-spline curve	0.97951	725.15104	45°1268′
		Proposed Algorithm	0.61845	709.13587	44°1534′
(400, 400)	(−400, −400)	A* algorithm combined with B-spline curve	1.24851	1634.9875	109°8418′
		Proposed Algorithm	0.75112	1568.27831	105°7518′
(400, −400)	(−400, 400)	A* algorithm combined with B-spline curve	1.190011	1612.83788	141°4894′
		Proposed Algorithm	0.74848	1567.15489	134°4537′

5.3. Simulation Experiment of Target Hunting

In order to verify the effectiveness of the proposed algorithm, the approaching method [35] was used to implement comparative experiments with the proposed algorithm. The approaching method is a method to directly plan the path of virtual structure points, and the virtual structure points were generated around the target. The target ship does not escape the hunting of multi-USV swarm in this simulation experiment. The experimental results of the approaching method are shown in Figures 19 and 20, and the experimental results of the proposed algorithm are shown in Figures 21 and 22—where the ordinate d of the curve in Figures 20 and 22 is the actual distance between the USV and the target calculated by Formula (24), and the abscissa is time. In this experiment, the multi-USV swarm consists of three ordinary ships, and the ordinary ships are called OS1, OS2, and OS3.

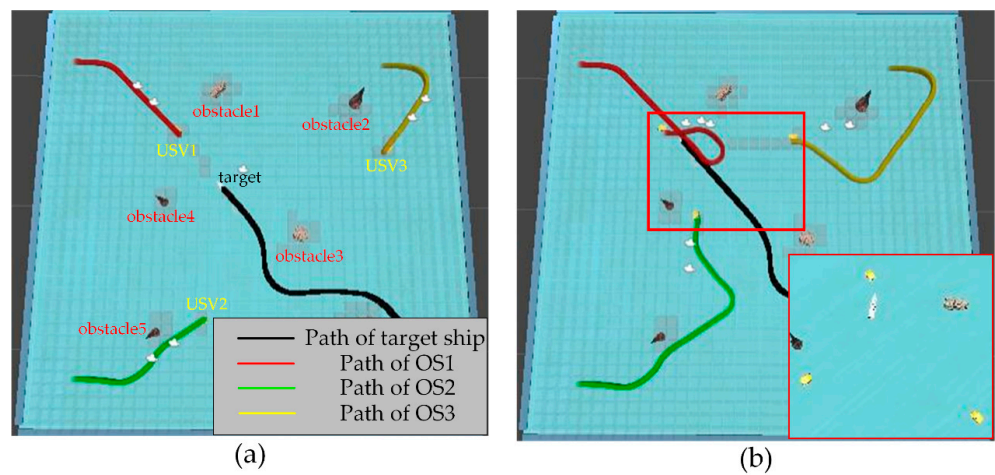


Figure 19. (a) The state of approaching method in the process of hunting. (b) The result of hunting by approaching method.

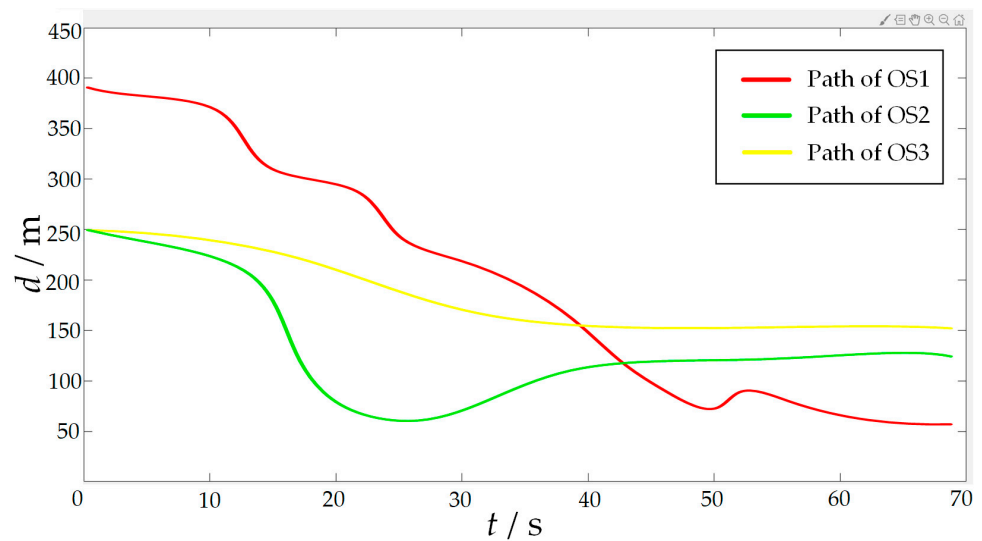


Figure 20. Distance change between the USV and target in the process of surrounding by approaching method.

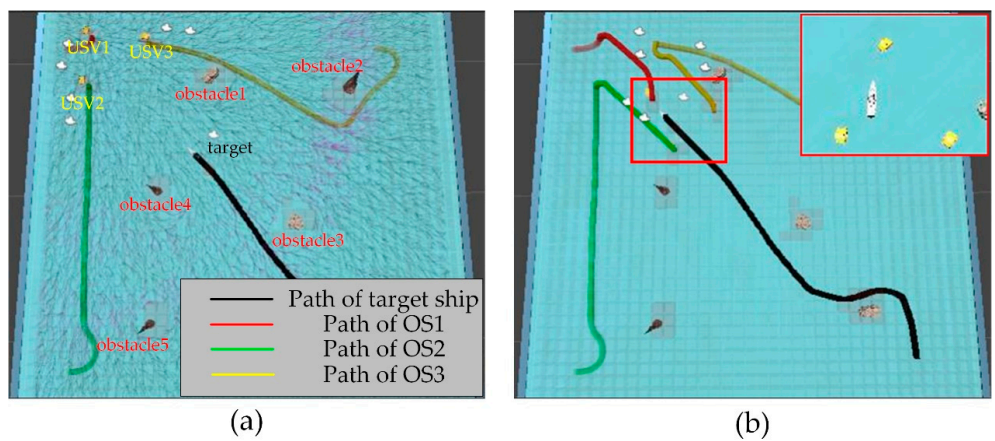


Figure 21. (a) The state of proposed algorithm in the process of hunting. (b) The result of hunting by proposed algorithm.

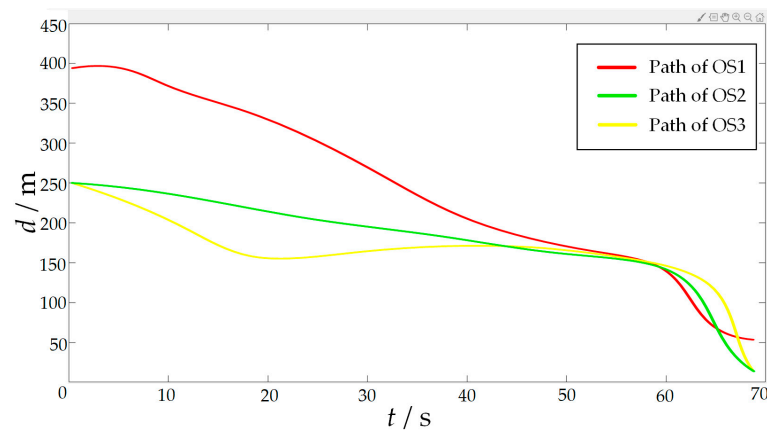


Figure 22. Distance change between USV and target in the process of surrounding by proposed algorithm.

In Figure 19a, it is shown that the multi-USV swarm was directly approaching the target ship. Although this can reduce the traveling distance of the USV, OS1 was closer to the target ship than OS2 and OS3. In Figure 19b, the result of target hunting is framed by red squares, it is shown that OS1 tracked the target ship and scraped with the target ship. In Figure 20, it is shown that OS2 and OS3 did not effectively encircle the target ship. The distance between the multi-USV swarm was too large, leading to the hunting failure. In Figure 21a, it is shown that the multi-USV swarm had generated a U-shaped array in front of the target ship, making preparations for hunting. In Figure 21b, the result of target hunting is framed by red squares; it is shown that the multi-USV swarm has surrounded the target ship. Because of the U-shaped array, the multi-USV swarm had formed an encirclement circle and successfully hunted the target ship. In Figure 22, it is shown that the USV with the proposed algorithm can maintain a similar distance from the target ship, and the formation of multi-USV swarm is more stable.

In addition, this study utilized the cooperative hunting method based on the Long short-term memory (LSTM) neural network and the cooperative hunting method based on artificial potential field (APF) as a comparative algorithm and conducted 10 comparative experiments with the proposed algorithm. The results are shown in Figure 23.

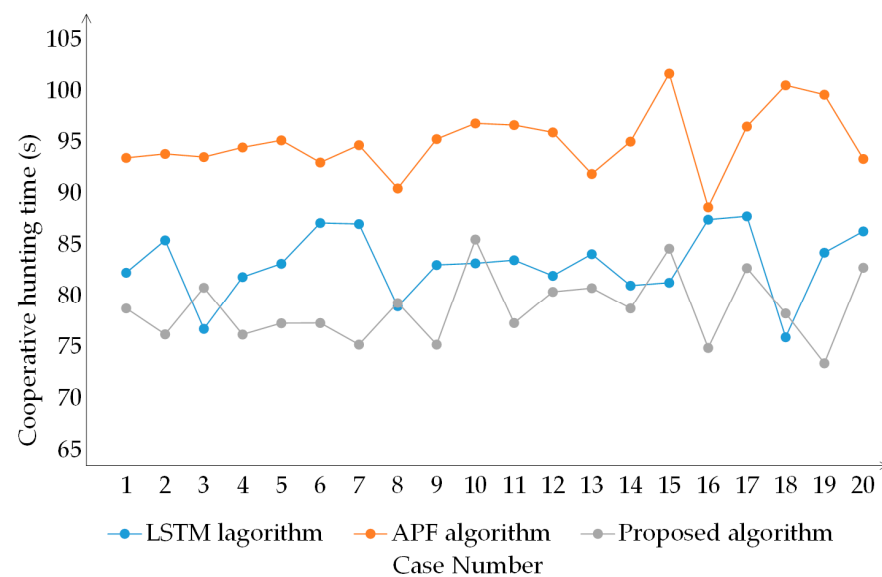


Figure 23. Multiple simulation experimental results of three algorithms.

From Figure 23, the performance of the proposed algorithm is far superior to the cooperative hunting method based on APF. At the same time, the proposed algorithm performs better than the cooperative hunting method based on LSTM in most cases. Although the cooperative hunting method based on LSTM outperformed the proposed algorithm in experiments 3, 8, 10, 15, and 18, it is within a reasonable range. In addition, the cooperative hunting method based on LSTM needs to be trained in advance and then applied to cooperative hunting, while the proposed algorithm does not require training. In summary, the proposed algorithm has high cooperative hunting efficiency.

In order to further verify the effectiveness of the proposed algorithm, escape action is added to the target ship. The target ship will always try to escape from the map boundary. At the same time, when the target ship finds another USV in the environment, it will stay away from the USV in the environment. The experimental results are shown in Figures 24 and 25. Where the ordinate d of the curve in Figure 25 is the actual distance between the USV and the target calculated by Formula (24), and the abscissa is time.

In Figure 24a, the start of target hunting is framed by red squares; it is shown that the multi-USV swarm had chased the target ship and formed an encirclement circle to surround the target. In Figure 24b, the process of target hunting is framed by red squares; it is shown that the target ship turned left to escape, and OS1 blocked the escape route of the target ship. In Figure 24c, the process of target hunting is framed by red squares; it is shown that the target ship had performed two escape actions, both of which were blocked by OS1. In Figure 24d, the result of target hunting is framed by red squares; it is shown that the multi-USV swarm had trapped the target ship in the encirclement circle and successfully controlled the target ship to prevent the target ship escaping from the map boundary. Therefore, the proposed algorithm can successfully implement encirclement when facing a target with a certain escape ability. In Figure 25, it is shown that the USV with proposed algorithm can maintain a similar distance from the target ship, and the formation of the multi-USV swarm is stable.

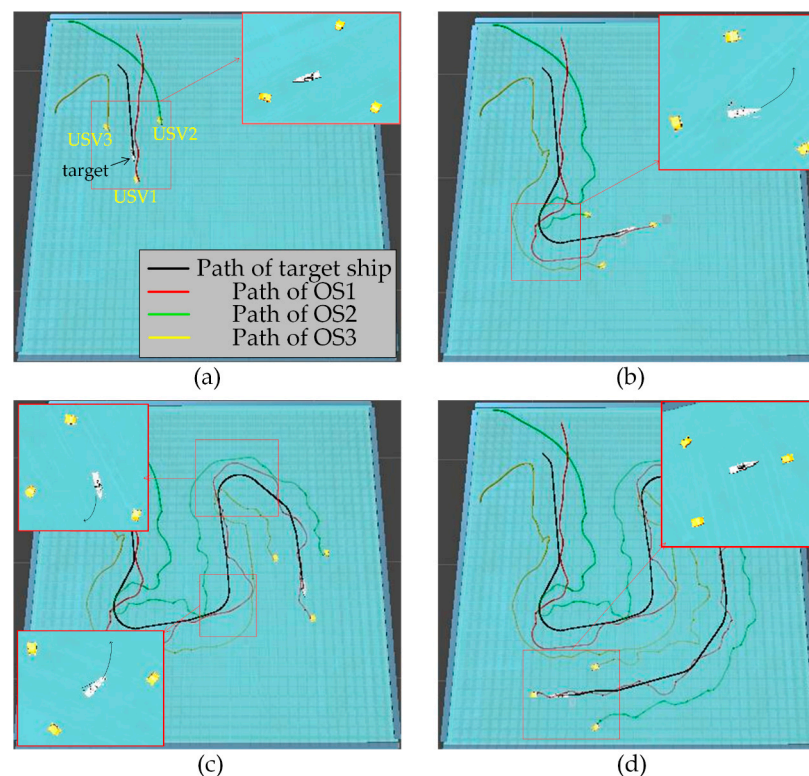


Figure 24. (a) One of the results of the hunting at the first moment. (b) One of the results of the hunting at the second moment. (c) One of the results of the hunting at the third moment. (d) One of the results of the hunting at the fourth moment.

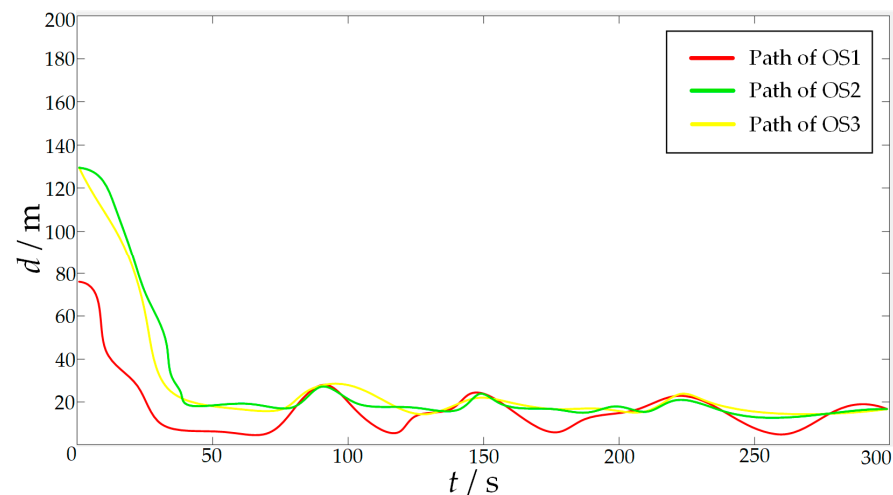


Figure 25. Distance change between the USV and target in the process of hunting by proposed algorithm.

6. Conclusions

This paper proposed a cooperative hunting method for multi-USV based on the A* algorithm in an environment with obstacles. First, on the basis of the traditional A* algorithm, a path smoothing method based on the minimum turning radius of USV is proposed. Second, a bionic based cooperative hunting method for multi-USV swarm is proposed to hunt the target. In order to verify the effectiveness of the proposed algorithm, two groups of simulation experiments are implemented. The results show that the proposed algorithm has good effectiveness in path planning and target hunting. The efficiency of cooperative hunting methods based on virtual structural points has been improved, and compared with comparative algorithms, the proposed algorithm can more effectively hunt targets.

This study did not consider the impact of ocean currents on USVs, and the impact of ocean currents will also be considered during the cooperative hunting in future research. In addition, in future research, the proposed algorithm in this paper can be improved in two aspects. First, most of the parameters in this paper are obtained from experience. In the future, the proposed algorithm will be improved with a depth learning algorithm to optimize its parameters. Second, this paper does not study the task allocation method, and the related algorithms will be studied to make reasonable task allocation for the multi-USV swarm in the future.

Author Contributions: Conceptualization, Z.C. and J.Y.; methodology, Z.C.; software, Z.C.; validation, J.Y., Z.Z. and X.W.; formal analysis, Y.L.; investigation, Z.C.; resources, J.X.; data curation, J.Y.; writing—original draft preparation, J.Y.; writing—review and editing, X.W.; visualization, X.W.; supervision, Z.Z.; project administration, J.Y.; funding acquisition, Z.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Program of China, grant number 2022YFF1101103. National Key Research and Development Program of China, grant number 2020YFC1606801. Beijing Municipal Natural Science Foundation, grant number 4222042.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created.

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

References

1. Yu, J.; Liu, G.; Xu, J.; Zhao, Z.; Chen, Z.; Yang, M.; Wang, X.; Bai, Y. A Hybrid Multi-Target Path Planning Algorithm for Unmanned Cruise Ship in an Unknown Obstacle Environment. *Sensors* **2022**, *22*, 2429. [[CrossRef](#)] [[PubMed](#)]
2. Yu, J.; Deng, W.; Zhao, Z.; Wang, X.; Xu, J.; Wang, L.; Sun, Q.; Shen, Z. A Hybrid Path Planning Method for an Unmanned Cruise Ship in Water Quality Sampling. *IEEE Access* **2019**, *7*, 87127–87140. [[CrossRef](#)]
3. Chen, Z.; Yu, J.; Zhao, Z.; Wang, X.; Chen, Y. A Path-Planning Method Considering Environmental Disturbance Based on VPF-RRT*. *Drones* **2023**, *7*, 145. [[CrossRef](#)]
4. Zhu, Z.; Xiao, J.; Li, J.; Wang, F.; Zhang, Q. Global path planning of wheeled robots using multi-objective memetic algorithms. *Integr. Comput. Aided Eng.* **2015**, *22*, 387–404. [[CrossRef](#)]
5. Altunbas, C.; Alexeev, T.; Miften, M. Effect of grid geometry on the transmission properties of 2D grids for flat detectors in CBCT. *Phys. Med. Biol.* **2019**, *64*, 225006. [[CrossRef](#)]
6. Zhang, Q.; Song, X.; Yang, Y.; Haotian, M.; Ryosuke, S. Visual graph mining for graph matching. *Comput. Vis. Image Underst.* **2019**, *178*, 16–29. [[CrossRef](#)]
7. Zhang, J.; Feng, Y.; Shi, F.; Wang, G.; Ma, B.; Li, R.; Jia, X. Vehicle routing in urban areas based on the oil consumption weight-Dijkstra algorithm. *IET Intell. Transp.* **2016**, *10*, 495–502. [[CrossRef](#)]
8. Yershov, D.; Lavalle, S. Simplicial Dijkstra and A* Algorithms: From Graphs to Continuous Spaces. *Adv. Robot.* **2012**, *26*, 2065–2085. [[CrossRef](#)]
9. Ni, J.; Wu, L.; Shi, P.; Yang, S. A dynamic bioinspired neural network based real-time path planning method for autonomous underwater vehicles. *Comput. Intell. Neurosci.* **2017**, *2017*, 9269742. [[CrossRef](#)]
10. Wang, P.; Gao, S.; Li, L.; Sun, B.; Cheng, S. Obstacle Avoidance Path Planning Design for Autonomous Driving Vehicles Based on an Improved Artificial Potential Field Algorithm. *Energies* **2019**, *12*, 2342. [[CrossRef](#)]
11. Ye, B.; Tang, Q.; Yao, J. Collision-Free Path Planning and Delivery Sequence Optimization in Noncoplanar Radiation Therapy. *IEEE Trans. Cybern.* **2019**, *49*, 42–55. [[CrossRef](#)]
12. Heon-Cheol, L.; Touahmi, Y.; Beom-Hee, L. Grafting: A Path Replanning Technique for RaPIly-Exploring Random Trees in Dynamic Environments. *Adv. Robot.* **2012**, *26*, 2145–2168.
13. Wanna, P.; Wongthanavas, S. An Improved Cellular Automata-Based Classifier with Soft Decision. *J. Internet Technol.* **2020**, *21*, 1701–1715. [[CrossRef](#)]
14. Zeng, M.; Xi, L.; Xiao, A. The free step length ant colony algorithm in mobile robot path planning. *Adv. Robot.* **2016**, *30*, 1509–1514. [[CrossRef](#)]
15. Thi-Thoa, M.; Cosmin, C.; Duc-Trung, T.; Robin, D. A Hierarchical Global Path Planning Approach for Mobile Robots Based on Multi-Objective Particle Swarm Optimization. *Appl. Soft Comput.* **2017**, *59*, 68–76. [[CrossRef](#)]
16. Jiang, Z.; Jun, W.; Xiao, S. Autonomous land vehicle path planning algorithm based on improved heuristic function of A-Star. *Int. J. Adv. Robot. Syst.* **2021**, *18*, 1–10. [[CrossRef](#)]
17. Yu, J.; Hou, J.; Chen, G. Improved Safety-First A-Star Algorithm for Autonomous Vehicles. In Proceedings of the 2020 5th International Conference on Advanced Robotics and Mechatronics (ICARM), Shenzhen, China, 18–21 December 2020.
18. Lin, Z.; Zhang, Y.; Li, Y. Mobile Robot Path Planning Based on Improved Localized Particle Swarm Optimization. *IEEE Sens. J.* **2021**, *21*, 6962–6972. [[CrossRef](#)]
19. Liu, Z.; Liu, H.; Lu, Z.; Zheng, Q. A Dynamic Fusion Pathfinding Algorithm Using Delaunay Triangulation and Improved A-star for Mobile Robots. *IEEE Access* **2021**, *9*, 20602–20621. [[CrossRef](#)]
20. Hong, Z.; Sun, P.; Tong, X.; Pan, H.; Zhou, R.; Zhang, Y.; Han, Y.; Wang, J.; Yang, S.; Xu, L. Improved A-Star Algorithm for Long-Distance Off-Road Path Planning Using Terrain Data Map. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 785. [[CrossRef](#)]
21. Zhang, Z.; Wu, J.; Dai, J.; He, C. Optimal path planning with modified A-Star algorithm for stealth unmanned aerial vehicles in 3D network radar environment. *Proc. Inst. Mech. Eng. Part G J. Aerosp. Eng.* **2021**, *236*, 72–81. [[CrossRef](#)]
22. Zhao, Z.; Hu, Q.; Feng, H.; Feng, X.; Su, W. A Cooperative Hunting Method for Multi-AUV Swarm in Underwater Weak Information Environment with Obstacles. *J. Mar. Sci. Eng.* **2022**, *10*, 1266. [[CrossRef](#)]
23. Chen, Y.; Ma, X.; Bai, G.; Sha, Y.; Liu, J. Multi-autonomous underwater vehicle formation control and cluster search using a fusion control strategy at complex underwater environment. *Ocean Eng.* **2020**, *216*, 108048. [[CrossRef](#)]
24. Wang, Y.; Dong, L.; Sun, C. Cooperative Control for Multi-Player Pursuit-Evasion Games with Reinforcement Learning. *Neurocomputing* **2020**, *412*, 101–114. [[CrossRef](#)]
25. Cai, L.; Sun, Q. Multiautonomous underwater vehicle consistent collaborative hunting method based on generative adversarial network. *Int. J. Adv. Robot. Syst.* **2020**, *17*, 663–678. [[CrossRef](#)]
26. Aggarwal, S.; Kumar, N. Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. *Comput. Commun.* **2019**, *149*, 270–299. [[CrossRef](#)]
27. Wang, Q.; Li, J.; Yang, L.; Yang, Z.; Li, P.; Xia, G. Distributed Multi-Mobile Robot Path Planning and Obstacle Avoidance Based on ACO-DWA in Unknown Complex Terrain. *Electronics* **2022**, *11*, 2144. [[CrossRef](#)]
28. Guo, J.; Qi, J.; Wang, M.; Wu, C.; Xu, S. A cooperative search and encirclement algorithm for quadrotors in unknown areas. *J. Beijing Univ. Aeronaut. Astronaut.* **2022**. [[CrossRef](#)]
29. Souza, C.; Newbury, R.; Cosgun, A.; Castillo, P.; Vidolov, B.; Kulic, D. Decentralized Multi-Agent Pursuit Using Deep Reinforcement Learning. *IEEE Robot. Autom. Lett.* **2021**, *6*, 4552–4559. [[CrossRef](#)]

30. Sun, Z.; Sun, H.; Li, P.; Zou, J. Self-Organizing Cooperative Pursuit Strategy for Multi-USV with Dynamic Obstacle Ships. *J. Mar. Sci. Eng.* **2022**, *10*, 562. [[CrossRef](#)]
31. Lv, J.; Xu, X.; Du, S.; Ma, Q. Research on the Method of Capturing Task Allocation Based on Energy Balance. In Proceedings of the 2nd International Conference on Artificial Intelligence, Network and Information Technology, Shanghai, China, 14–15 October 2021.
32. Ammar, A.; Bennaceur, H.; Chari, I.; Koubâa, A.; Alajlan, M. Relaxed Dijkstra and A* with linear complexity for robot path planning problems in large-scale grid environments. *Soft Comput.* **2016**, *20*, 4149–4171. [[CrossRef](#)]
33. Liu, L.; Han, G.; Xu, Z.; Jiang, J.; Martinez-Garcia, M. Boundary Tracking of Continuous Objects Based on Binary Tree Structured SVM for Industrial Wireless Sensor Networks. *IEEE Trans Mob. Comput.* **2020**, *21*, 849–861. [[CrossRef](#)]
34. Yu, J.; Chen, Z.; Zhao, Z.; Yao, P.; Xu, J. A traversal multi-target path planning method for multi-unmanned surface vessels in space-varying ocean current. *Ocean. Eng.* **2023**, *278*, 114423. [[CrossRef](#)]
35. Xie, Y.; Liang, X.; Lou, L.; Guo, X. Self-organization Method of USV Swarm Target Strike Task Based on Ant Colony Algorithm. In Proceedings of the International Symposium on Autonomous Systems Systems Engineering Research Institute, CSSC, Beijing, China, 29–31 May 2019.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.