Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/mbe

MBE, 21(2): 2302–2322.
DOI: 10.3934/mbe.2024101
Received: 15 November 2023
Revised: 25 December 2023
Accepted: 09 January 2024
Published: 12 January 2024

Research article

An enhanced A* method incorporating an encrypted memory database for ASV efficient local path planning

Yuanshuo Liu[1], **Defeng Wu**[1,2,*] **and Zheng You**[1,2]

[1] School of Marine Engineering, Jimei University, Xiamen, China

[2] Fujian Provincial Key Laboratory of Naval Architecture and Ocean Engineering, Xiamen 361021, China

* **Correspondence:** Email: defeng@jmu.edu.cn.

Abstract: For the autonomous surface vehicle (ASV) planning problem, an enhanced A* method incorporating encrypted memory database for ASV efficient local path planning is proposed. Considering the current various path planning problems mostly use methods with high time complexity, such as neural networks, we select the A* algorithm with low time complexity as the basis. To speed up the path planning rate and further improve the real-time and realistic algorithm, this paper modifies the heuristic function of the A* algorithm by combining the motion mode of ASV. In response to the problem that the target point is far from the detection, we improve the target point design mechanism and create a new temporary target point within the detection range. In addition, the algorithm incorporates a memory database, which can record commonly used waters or retain the environmental path of navigated waters as a priori information. When the same waters are reencountered, the memory database information can be read directly to complete the navigation. Moreover, the memory database is encrypted to prevent information leakage. Finally, a simulation environment is built to verify the effectiveness of the proposed algorithm by comparison with some existing algorithms.

Keywords: autonomous surface vehicle (ASV); path planning; memory database; enhanced A* algorithm; improved heuristic function

1. Introduction

In recent decades, the advancement of artificial intelligence research and the gradual maturity of unmanned technology both contribute to the enormous leap in unmanned systems [1–3]. As a multidisciplinary and comprehensive subject involving information technology, automation technology and system integration, unmanned technology has been widely utilized in drones [4, 5], autonomous surface vehicles (ASVs) [6, 7], flexible spacecrafts [8] and undersea robots [9].

Particularly, ASVs can replace humans to accomplish more complex and dangerous tasks in many fields: In the civilian field, ASV is widely used in meteorology [10], surveying and mapping [11, 12], environmental protection [13], vessel train [14], and agriculture [15]; in military field, in addition to tactical simulation as a target [16], ASVs can also be used as enemy situation reconnaissance [17], communication relay [18] and effectiveness assessment [19]. The above tasks require ASVs to have higher autonomous capabilities, and the mission process's completion needs to consider actual constraints such as the shortest route and lowest loss [20]. Therefore, as the basis for completing the requirements, path planning plays a crucial role in the ASV system.

In general, path planning problems can be categorized as follows: traversal optimal path planning within discrete domains, including the traveling merchant problem [21] and its derivative issues; Shortest path planning within discrete domains, such as road network and routing problems; Traversal path planning in the continuous domain is mainly applied to cleaning robots [22, 23], weeding robots [24], search and rescue robots [25], mineral detectors [26]. Path planning problems within the continuous domain are divided into global planning, which focuses on path optimization, and local planning, which focuses on safety. Global path planning avoids known solid obstacles based on prior knowledge of the environment to find the optimal path, and typical application scenarios include autonomous movement planning of robotic arms [27], path planning of drones [28], and trajectory planning of aerodynamic missiles [29]. However, it should be noted that the above literature is ideal because there are always unknown or dynamic obstacles in the navigation environment of the ASV system. It should be mentioned that the local planning algorithm can successfully overcome the limitations caused by global planning. As a real-time, efficient, and stable algorithm, the local planning algorithm can effectively use dynamic and actual environmental information and achieve timely obstacle avoidance. As a consequence, it is often necessary to model the environment using methods such as the grid method [30, 31].

Standard path planning algorithms can be divided into geometric algorithms, bionic algorithms, and intelligent algorithms [32, 33] according to their fundamental principles. For geometric algorithms, Yu [34] proposed an artificial potential field RRT* algorithm. The artificial potential field for sampling algorithm and the ocean current constraint function are implemented in the RRT* algorithm, and the modified Hungarian algorithm for multiobjective allocation is used; For bionic algorithms, Escario [35] used the swarm algorithm to address the path planning problem of ASVs, aiming to obtain optimal trajectories for kinematic of ASVs; for intelligent algorithms, Singh [36] introduced artificial potential fields for ASV path planning in real-time marine environments. Path cost and its length, calculation time are described to guarantee the effective planning of pathways.

In summary, path planning is necessary for ASVs to perform tasks. However, some of the existing algorithms have disadvantages such as slow exploration speed, repeated exploration, leakage of crucial information, and they are limited by the detection capabilities of ASVs themselves, which cannot be applied to application scenarios where the target point is too far away. For example, the deep learning algorithm needs a large amount of data training; the ant colony algorithm and the artificial potential field method need to carefully adjust the parameters; the genetic algorithm has slow convergence speed and is easy to falls into local optimum. To overcome these limitations, this paper chooses the A* algorithm which has simple structure but similar effect. Here, we proposed a local path planning method for ASVs based on an enhanced A* algorithm. First, an improved heuristic function is incorporated into the traditional A* algorithm to speed up exploration, and the

obstacle-related function is added to increase path safety. Then, a temporary target within the detection range is created to improve the target point mechanism when the target point is too far to be detected. Besides, the additional memory function is introduced to store the frequently used or has been recorded water. When the same water area is reencountered, the corresponding path can be directly read and complete the voyage without the need to search for the path again. Furthermore, the memory database is encrypted to prevent information leakage. Finally, the feasibility and superiority of the simulation environment are verified.

2. The ASV system

2.1. ASV model

Based on the existing ship modeling theory, the mathematical model widely used has been adopted [37], which is proposed by Fossen. The following assumptions need to be satisfied before modeling:

(1) The overall mass of the ASV is evenly distributed;
(2) ASVs are symmetrical left and right;
(3) Ignoring the influence of Earth's rotation.

The motion coordinate system is the basis for analyzing the motion of ASVs. As shown in Figure 1(a), an inertial coordinate system XOY and a coordinate system attached to the ASV $X_bO_bY_b$ are defined. Somewhere on the Earth's surface is chosen as the coordinate origin O of the inertial coordinate system, with the OX axis pointing towards the north and the OY axis pointing towards the east. When the ship is symmetrical, the ship's center of gravity O_b is selected as the origin of the coordinate system attached to the ASV, with the axis O_bX_b pointing to the bow along the ship's centerline and the axis O_bY_b pointing to the port side of the ship.

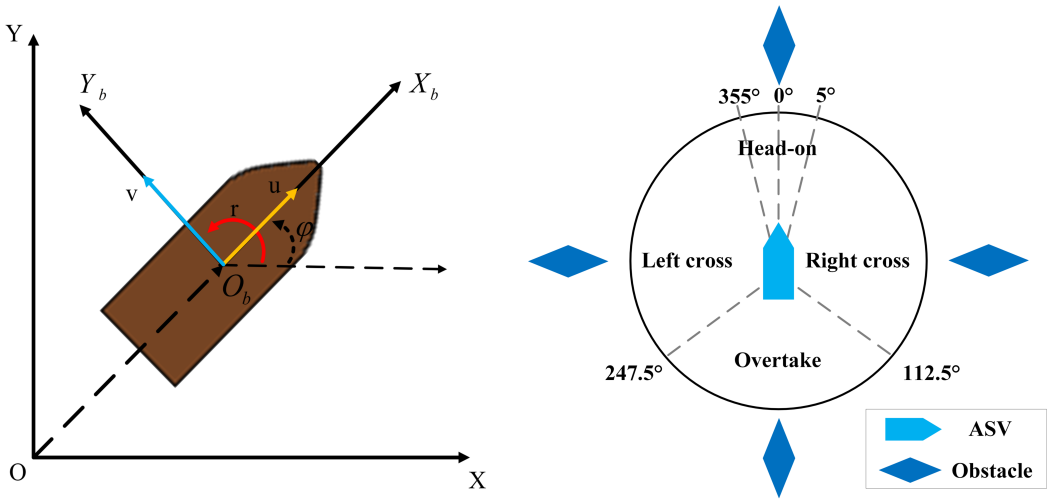

Figure 1. ASV system.

To simplify the analysis, the ASV's roll, pitch, and heave are usually ignored, and only the plane motion of the ASV is considered. The kinematic model can be described as.

$$\begin{cases} \dot{x} = u\cos\psi - v\sin\psi \\ \dot{y} = u\sin\psi + v\cos\psi \\ \dot{\psi} = r \end{cases} \tag{2.1}$$

where $\eta = [x, y, \psi]^T$ represents the position vector of the ASV in the inertial coordinate system, consisting of position (x, y) and bow angle $\psi \in [0, 2\pi]$. $v = [u, v, r]^T$ represents the velocity vector in the coordinate system attached to the ASV, consisting of longitudinal velocity u, lateral velocity v, and yaw angular velocity r. By defining the rotation matrix $M(\psi)$, which is shown as Eq (2.2).

$$M(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

Eqs (2.1) and (2.2) can be simplified to obtain Eq (2.3).

$$\dot{\eta} = M(\psi)\, v \tag{2.3}$$

Moreover, the corresponding ASV dynamics model can be expressed as Eq (2.4).

$$A\dot{v} = T - S(v)v - Lv + M(\psi)^T p \tag{2.4}$$

In Eq (2.4), $A = \begin{bmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & a_{23} \\ 0 & a_{23} & a_{33} \end{bmatrix}$ is a system inertia matrix; $S(v) = \begin{bmatrix} 0 & 0 & s_{13} \\ 0 & 0 & s_{23} \\ s_{31} & s_{32} & 0 \end{bmatrix}$ is the Coriolis and centripetal force matrix; $L = \begin{bmatrix} l_{11} & 0 & 0 \\ 0 & l_{22} & l_{23} \\ 0 & l_{32} & l_{33} \end{bmatrix}$ is a linear damping matrix; $T = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$ is a kinematic matrix including longitudinal control t_1, lateral control t_2, and yaw angle control t_3; $p = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$ stands for the unknown disturbance caused by wind, waves and ocean currents, and the value of p is assumed as constant or slowly changing.

2.2. ASV obstacle avoidance rules

In 1920, the International Maritime Organization proposed a set of maritime traffic rules to regulate ship traffic, the International Regulations for Preventing Collisions at Sea (COLREGS). This set of rules governs the navigation of ships piloted by humans and defines the different situations of ship traffic. Moreover, it has been updated in 1972 [38]. Although it is unreasonable for ASVs to sail entirely according to this scheme, it has a certain reference value. There are several important provisions in the COLREGS:

Article 13 of the COLREGS, overtaking:

(1) When a ship overtakes another ship from a direction greater than 22.5 degrees directly behind another ship, i.e., the position of the ship it is overtaking, and at night it can see only the stern

lights of the overtook ship, it cannot see any of the overtook ship's sidelights, it shall be considered to be in overtaking.

(2) When a ship has any doubt as to whether it is overtaking another ship, the ship shall assume it is overtaking another ship and act accordingly.

(3) Any subsequent change in the orientation between the two ships shall not treat the overtaking ship as a cross ship within the meaning of the various articles of the COLREGS or relieve it of its responsibility to yield the overtook ship until it has finally passed through the region.

Article 14 of the COLREGS, head-on:

(1) When two ships meet in opposite or nearly opposite directions and pose a risk of collision, they shall turn to the right and each passes over the port side of the other ship.

(2) When a ship sees another ship directly in front or the near front, and at night, only visible the masthead lights of another ship in a straight line or close to a straight line, and so are two sidelights; in the daytime, when the corresponding form of another ship is seen like above, the head-on situation should be considered to exist.

(3) When a ship doubts the existence of the head-on situation, the ship shall assume that the situation does exist and act accordingly.

Article 15 of the COLREGS, crossing:

When two ships come across each other and pose a risk of collision, the ship with another ship on its starboard side shall give way to another ship, and if circumstances permit, it shall avoid crossing the front of the other ship.

Different navigation situations are divided based on the above rules of COLREGS, and the basic obstacle avoidance model of ASVs, as shown in Figure 1(b), is established.

2.3. Encryption algorithm

With the maturity of information technology, computer networks have become one of the essential means for human beings to disseminate information. The ASVs share data and transmit messages through the web to bring great convenience and opportunities for navigation. However, as an open and shared environment, the network environment has various factors that affect security issues, resulting in data interception, theft, and tampering. Therefore, this article encrypts the map of ASV paths database to improve the security of data transmission in the network environment. As a result, the leakage of the map database with malicious third parties intercepting and attacking can be prevented, which contributes to the economy and security of the ASVs. Current encryption technology can be divided into symmetric encryption and asymmetric encryption according to its fundamental type, and the two encryption flowcharts are shown in Figure 2(a).

Asymmetric encryption is irreversible, as shown on the left of Figure 2(a). Its encryption key A is public, and anyone can use this encryption key A to encrypt data. However, the encrypted information can only be unlocked by the corresponding decryption key B, which differs from key A, and key B is not public. Symmetric encryption, on the other hand, is reversible in encryption and decryption uses the same key, meaning that the sender and receiver of the message must hold the same key. As an open algorithm, it has small computation, fast encryption speed, and high efficiency. To conform with ASV's small memory and low computing power, this paper adopts a simple method of symmetric encryption.

Furthermore, the planned path is stored in the memory database and encrypted with a random key to increase security. The encryption and decryption calculation formula is as Eq (2.5).

$$\begin{cases} R_i = Cal(L_i, m) \\ L_i = -Cal(R_i, m) \end{cases} \tag{2.5}$$

where L_i is the data of the encrypted map position i; R_i is the decrypted map position i; m is the key matrix; Cal is the decryption formula, which is set in this article as the addition and subtraction set of values.

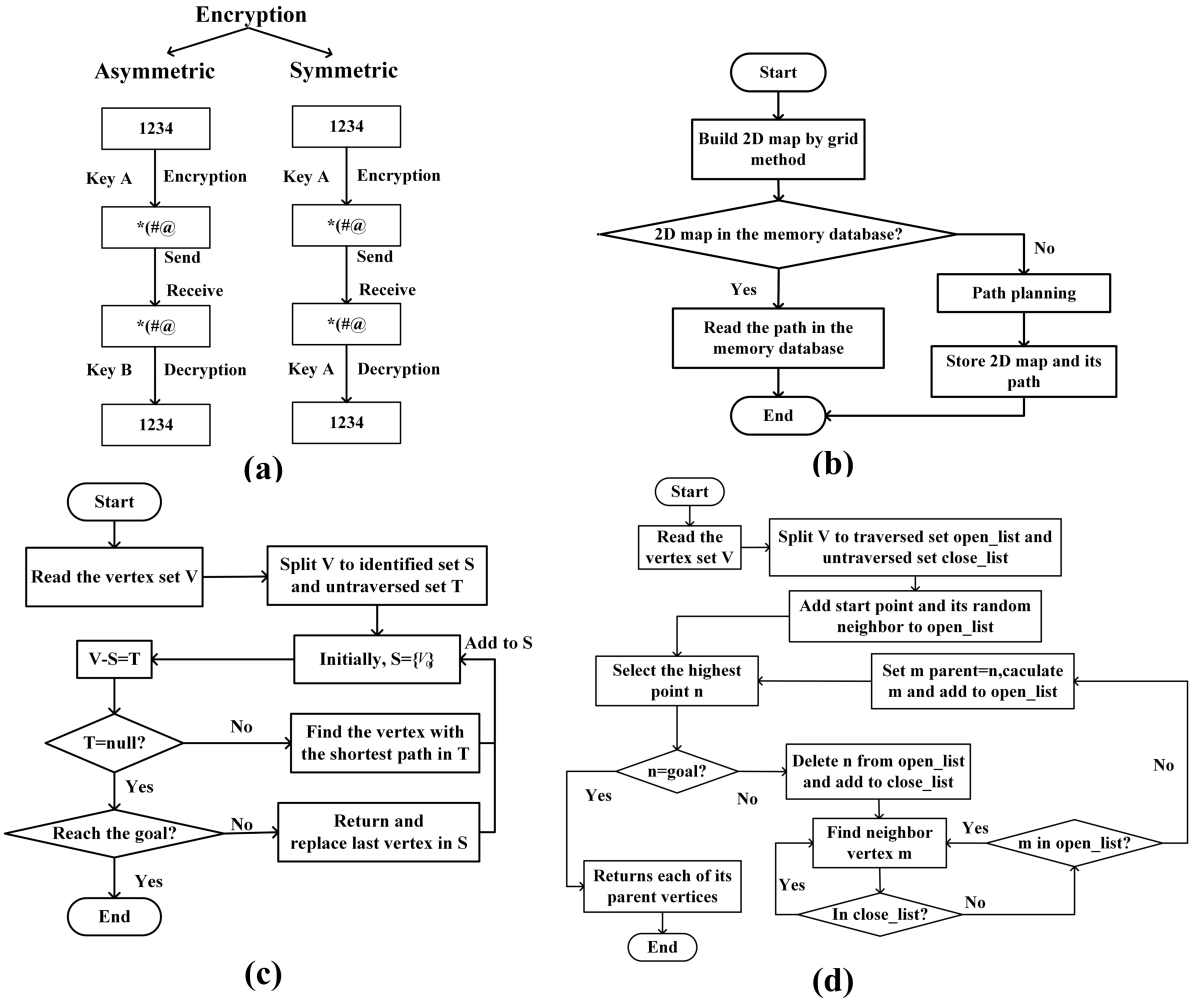

Figure 2. Flow chart.

3. Path planning algorithm

3.1. Dijkstra's algorithm

The Dijkstra's algorithm, an algorithm for shortest paths from one point to hlanother point was proposed in 1959 by Dijkstra, a Dutch computer scientist. Dijkstra's algorithm is designed to find the

shortest path in weight graph [39]. Based on the greedy algorithm, Dijkstra's algorithm starts from the starting point, and then each time traverses the not-visited neighboring nodes that are closest to the starting point until it extends to the final target point. The algorithm flow is shown in Figure 2(c).

An undirected graph $G = \{V, E\}$ is built, where V is the set of vertices in G, and E is the set of arcs. Initially, let $S = \{V_0\}$, $T = V - S$. If the distance value corresponding to the vertex T exists, $d(V_0, V_i)$ is set to the weight on the arc; If it does not exist, $d(V_0, V_i) = \infty$. Select the vertex with the smallest value from T connected to the vertex in S. If the vertex's weight is more minor than that has been used as the middle vertex and the distance value from V_0 to V_i is shortened, the previously used vertex will be modified. In addition, repeat the above steps until S contains all vertices.

When Dijkstra's algorithm searches for paths, it needs to expand from the starting point to all surrounding nodes and obtain all paths to the target point. Then the optimal way can be achieved. The path obtained by Dijkstra's algorithm must tend to be optimal, but it needs to expand many useless nodes when searching for the optimal way. Therefore, Dijkstra's algorithm is resource-consuming and low efficiency, especially in the case of a large map environment.

3.2. A algorithm*

The A* algorithm, published in 1968 by Peter Hart of the Stanford Institute [40], can be considered an expansion of Dijkstra's algorithm. As a heuristic algorithm, the A* algorithm works by using heuristic information to find the best path to travel, which searches nodes in the map and sets the appropriate heuristic function. Through estimating each node's value, the optimal node for the next expansion is obtained unless it achieves the final target point. A* algorithm is one of the commonly used path finding and graph traversal algorithms, with good performance and accuracy, and its algorithm flow is shown in Figure 2(d).

When running the A* algorithm, the best and value-optimal node which is selected from the priority queue each time is the first node to be explored. Besides, the A* algorithm has two sets of lists, namely open-list, representing the nodes that should be traversed, and close-list for the nodes that have been traversed, respectively. The A* algorithm calculates the priority of each node through Eq (3.1).

$$f(n) = g(n) + h(n) \tag{3.1}$$

where $f(n)$ is the comprehensive prioritization of node n; $g(n)$ is the charge of node n from the starting point; $h(n)$ is the estimated charge of node n from the final target point, also called the heuristic function. When $h(n) = 0$ and $f(n) = g(n)$, the algorithm degenerates into the Dijkstra's algorithm.

The actual charge of node n to the final target point is assumed as $Z(n)$. If $h(n) < Z(n)$, then the A* algorithm ensures that the shortest path will always be found. However, the smaller the $h(n)$ value is, the more nodes the algorithm will traverse, and the slower the compute speed. If $h(n) = Z(n)$, the A* algorithm will get the optimal path, and the operation speed will be faster. However, $Z(n)$ cannot be accurately calculated in actual operation. If $h(n) > Z(n)$, then the A* algorithm does not always find the shortest path, but it is the fastest.

It can be concluded that the speed and accuracy of the A* algorithm can be controlled by adjusting the heuristic. In practical application, the path priority needs to be considered, different tasks require different weights, and the impact of the characteristics of ASVs is also significant. For example, the cost of different sailing states of ASVs needs to be considered when finding the shortest path. If

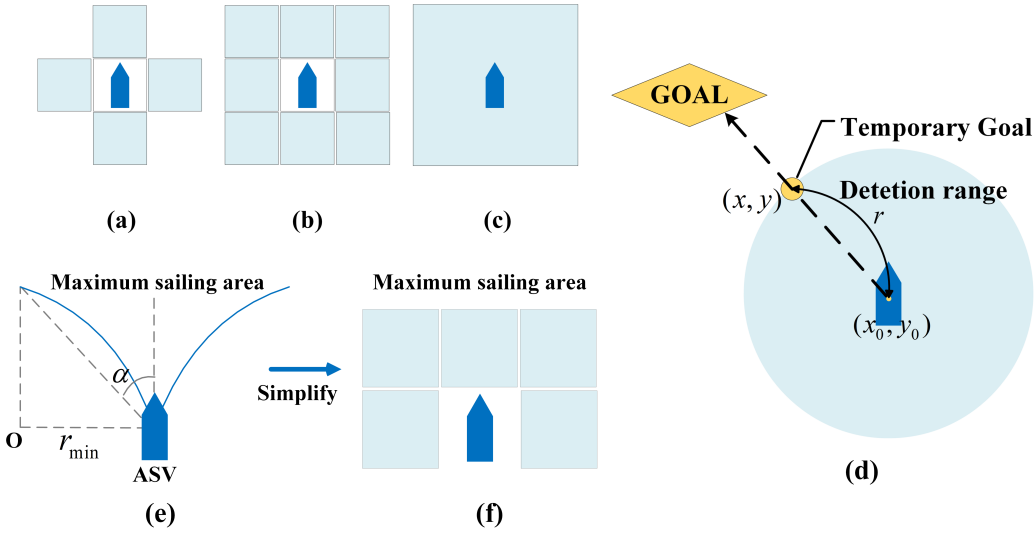

Figure 3. Movement of ASV under the grid map.

building the environment with a grid map, the distance between each point represents the loss, and there are three forms of motion, as shown in Figure 3(a)–(c).

If we allow only the ASV to move back and forth, left and right, as shown in Figure 3(a), the Manhattan distance is more accessible; If the eight directions shown in Figure 3(b) are permitted, then the Chebyshev distance is more appropriate; If ASV is allowed in any direction as shown in Figure 3(c), Euclidean distance can be used.

3.3. Enhanced A algorithm*

Based on the A* algorithm, this paper proposed an enhanced A* algorithm which improves the temporary target setting under extreme conditions, heuristic function and encrypted map database of the region have been explored.

In detail, if only partial information is available, for example, the approximate direction of the target point due to external or internal factors, the proposed enhanced A* algorithm can newly create a temporary target according to the approximate position of the final target point and the detection range of the ASV. The temporary target planning is shown in Figure 3(d), the calculation formula is as Eq (3.2).

$$\begin{cases} y - y_0 = k(x - x_0) \\ (y - y_0)^2 + (x - x_0)^2 = r^2 \end{cases} \tag{3.2}$$

where x_0 and y_0 are the current positions of the ASV; k is the orientation of the target point; r is the maximum radius that the ASV can detect; x and y are the coordinates of the temporary target of the ASV.

The A* algorithm exploration mode is improved, and the actual cost function is modified by using the lower limit of the turning radius, the lower restraint of the path length, and the constraint of the driving speed condition of the ASV. As a result, the obtained planned path is more in line with the kinematic model of the ASV, as shown in Figure 3(e),(f). $r_{\min}$ is the minimum turning radius with

center O of the ASV; α is the maximum angle of rotation, less than 35 [41]; and the two blue arcs form the maximum sailing area of the ASV.

By considering the simplified diagram of the kinematic model of the ASV above shown in Figure 3(f) and the relation between the ASV and the obstacle position, the A* algorithm with improved heuristic function is proposed to modify the obstacle avoidance problem during path planning, which contributes to the increase of safety and efficiency of navigation. The proposed heuristic function is defined as follows.

$$h(n) = |x - x_0| + |y - y_0| + \alpha C \tag{3.3}$$

where α is the discount parameter,$\alpha > 0$; and C is the relationship between the ASV and nearby obstacles, expressed by Euclidean distance in this paper.

To determine a better value of α, the algorithm in this paper randomly generates the initial environment in Figure 4 with pink obstacles and green start and finish points and carries out several tests. As shown in Table 1, when $\alpha = 0.1$, the algorithm performs better in terms of search time and running time. Therefore, 0.1 is chosen as the discount parameter.

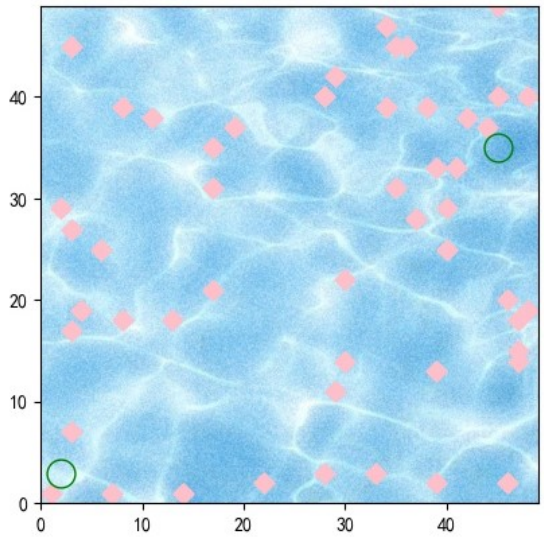

Figure 4. Environment for testing α.

Table 1. Comparison of different discount parameters α.

α	Search Times	Path Length	Running Time
0.1	160	44	26.98
0.3	320	44	60.01
0.5	932	44	169.08
0.8	1357	44	271.90

Besides, the memory function is considered to store the map data and their corresponding paths that have been obtained, and when the same map occurs in the future, the path can be directly read. Furthermore, the map database can be encrypted. The whole map storing and reading procedures are

shown in Figure 2(b). First, a two-dimensional (2D) map is built by the grid method. Then, compare the current 2D map with the map in memory database, and if they are identical, its corresponding path will be directly read as the voyage path; if they are not identical, a new path will be planned as the navigation route, and then the current 2D map will be saved with its corresponding navigation path in the memory database.

4. Simulation experiments

4.1. Environment establishment

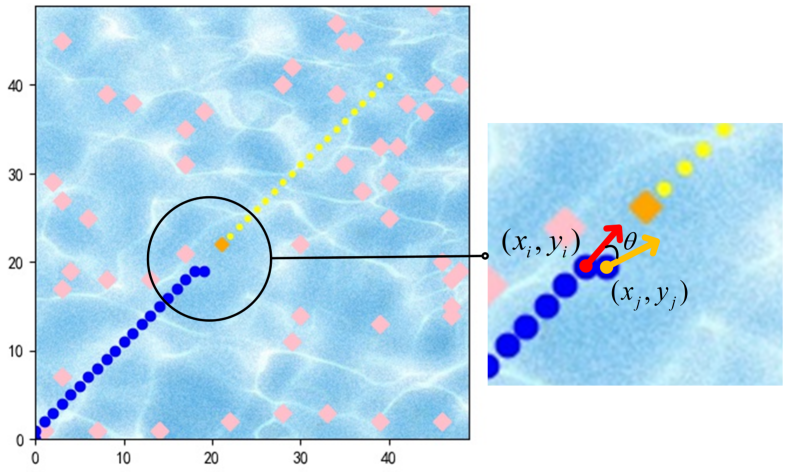

Figure 5. Initial map.

A grid map is built that scales the 2D environment and it is assumed each grid state is independent. For the group of grids in the environment, the value of grids without obstacles, i.e., available through grids should be assumed to be 0. If there is an obstacle at the grid, its value is set to non-0, and the solid obstacle is set to 1; the dynamic obstacle is set to 4; the dynamic obstacle's path is set to 3; and the algorithm planning path is set to 2.

By setting different colors for grids with different values, the solid obstacle's value 1 is set to pink; the dynamic obstacle's 4 is set to orange, and its path 3 is yellow; the algorithm planning path 2 is set to blue. Plotted by the python matplotlib function, the overall environment is shown in Figure 5(a).

4.2. Path search

Based on the above, the 2D map environment is constructed. The through regions are generated by combining the ASV kinematics model and COLREGS in Section 2. As shown in Figure 5(b) the region framed by the x-axis $(13, 28)$ and y-axis $(12, 27)$ in Figure 5(a) is enlarged. In terms of points (x_i, y_i) and (x_j, y_j), the red arrow is the moving direction of the ASV at point (x_i, y_i), while the yellow arrow is the moving direction of the ASV at point (x_j, y_j). The angle θ between the two arrows is the moving trend angle of the ASV when it moves from (x_i, y_i) to (x_j, y_j). Additionally, the moving trend angle θ is limited to the maximum turn angle of ASV, i.e., less than 35.

By combining the generated path and Eq (4.1), the moving trend angle θ between neighboring points (x_i, y_i) and (x_j, y_j) can be calculated via the following equation.

$$\theta = \arctan \frac{y_j - y_i}{x_j - x_i} \tag{4.1}$$

Table 2. The trend angle result.

	1	2	3	4	...	j	...
x	1	0	0	1	...	45	...
y	0	2	3	3	...	35	...
θ	−0.464	0.785	0	−0.785	...	−0.785	...

The trend angle result is shown in Table 2.

4.3. Encryption and decryption

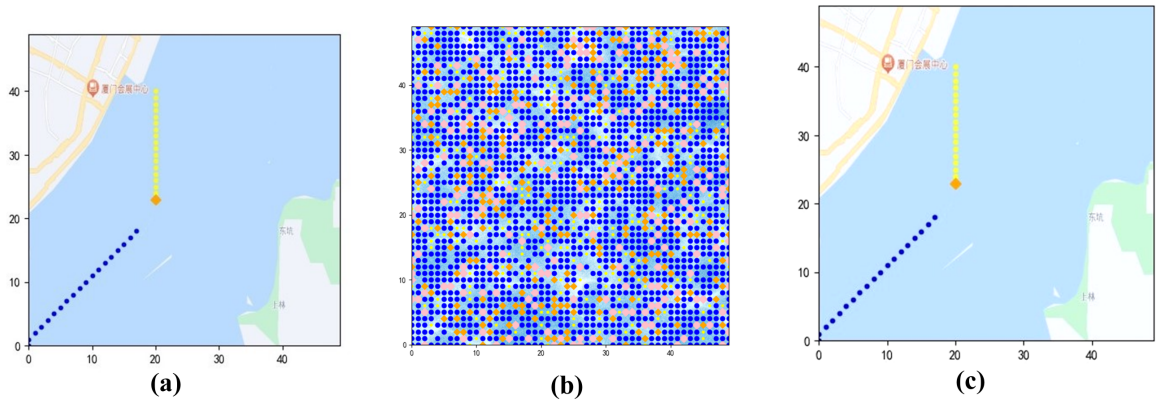

Figure 6. Path map encryption and decryption.

In this paper, the memory database is encrypted by symmetric encryption algorithms. If there is no correct key, the wrong path is obtained. Our experience can be shared with third parties to show the routes we have taken through the waters. The encrypted maps are protected from competitors and malicious third parties who may cause damage. For example, the path before encryption is shown in Figure 6(a), and it is encrypted using a symmetric encryption algorithm. The path map after encryption is shown in Figure 6(b), and the ASV path can no longer be distinguished. The path map after decryption is shown in Figure 6(c), where the path of the ASV can be easily observed.

4.4. Results comparison

In the following, the comparison among Dijkstra's algorithm, A* algorithm, the other A* algorithm in [42] and enhanced A* algorithm will be demonstrated. There are some indicators for the comparison: Whether the path can be planned; the length of planned path; the number of searches; the number of second searches; and the calculating time of algorithms. There are three experimental environments: The maze environment; the narrow water area, taking the water near the Garden Expo

Park in Xiamen City, Fujian Province, as an example; and the complex water area, like the area between Xiamen Island and Xiaojinmen Island.

4.4.1. Maze environment

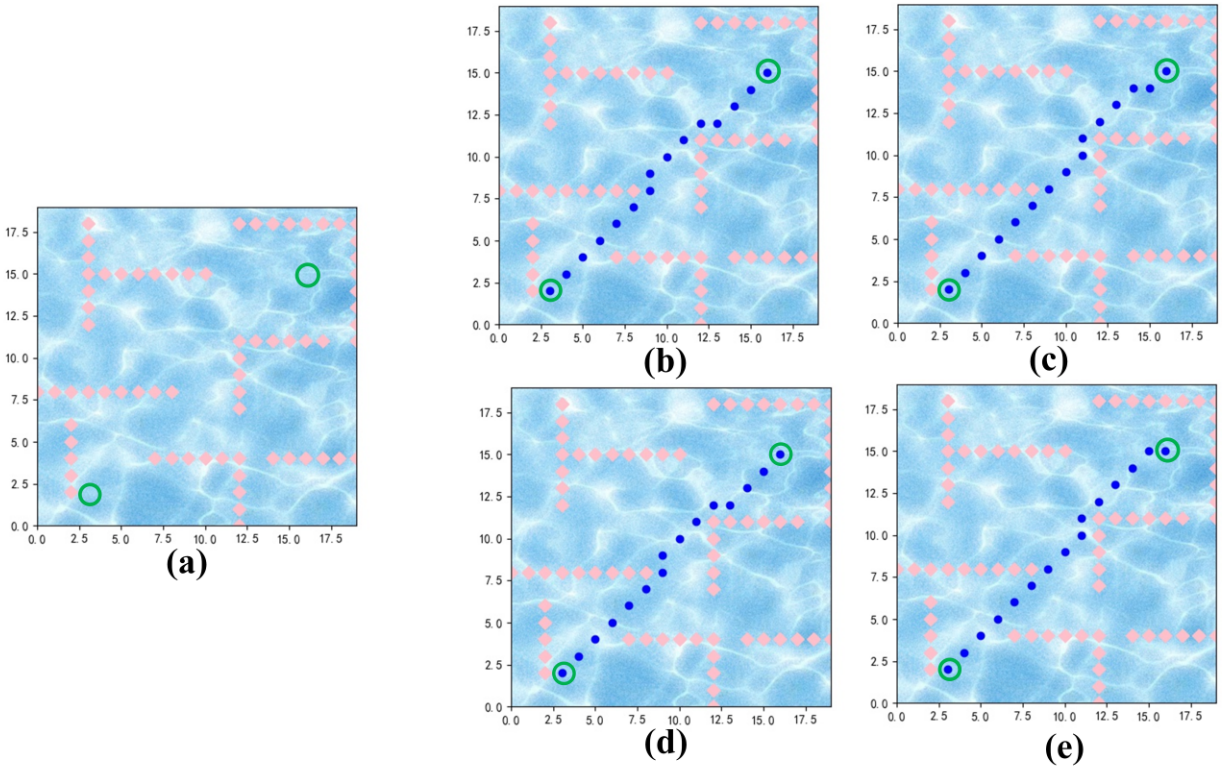

Figure 7. Maze environment.

Table 3. Algorithm comparison in a maze environment.

Algorithm	$h(n)$	Search	Second Search Times	Path Length Times	Time
Dijkstra	0	197	197	19.8	37.01
A*	$\sqrt{(x-x_0)^2+(y-y_0)^2}$	73	73	19.8	12.71
The other A* in [42]	$\alpha * \sqrt{(x-x_0)^2+(y-y_0)^2} + \beta * \arctan\frac{y-y_0}{x-x_0}$	101	101	19.8	17.44
Enhanced A*	$\lvert x-x_0\rvert + \lvert y-y_0\rvert + \alpha C$	51	0	14.0	8.83

First, the maze environment is built. As shown in Figure 7(a), the map size is 20×20. The green circle at the bottom of the map is used as the starting point with the coordinates $(3, 2)$, and the green circle at the top is the final target point with the coordinates $(16, 15)$.

Four algorithms are used for path planning, and the planning results are shown in Figure 7(b)–(e). The algorithms discussed in the previous were evaluated for their efficiency in finding the optimal path. The results showed that the enhanced A* algorithm, as shown in Figure 7(e), outperformed Dijkstra's algorithm, A* algorithm, and the other A* algorithm in [42], as shown in Figure 7(b)–(d), respectively. Specifically, the search times for the enhanced A* algorithm were 51 times faster than the search times

Table 4. Algorithm comparison in narrow environment.

Algorithm	$h(n)$	Search	Second Search Times	Path Length Times	Time
Dijkstra	0	599	599	80.61	116
A*	$\sqrt{(x-x_0)^2+(y-y_0)^2}$	336	336	80.61	62.78
The other A* in [42]	$\alpha * \sqrt{(x-x_0)^2+(y-y_0)^2} + \beta * \arctan\frac{y-y_0}{x-x_0}$	594	594	80.61	120.60
Enhanced A*	$\lvert x-x_0\rvert + \lvert y-y_0\rvert + \alpha C\mathrm{w}$	241	0	57.00	48.24

for Dijkstra's algorithm, A* algorithm, and the other A* algorithm in [42], which are 73 times, 101 times, and 197 times, respectively. Moreover, the enhanced A* algorithm also had the shortest path length and the calculating time, as shown in Table 4. Furthermore, the enhanced A* addition memory function can give this path without searching when reencountering the same map, effectively saving resources.

4.4.2. Narrow water environment

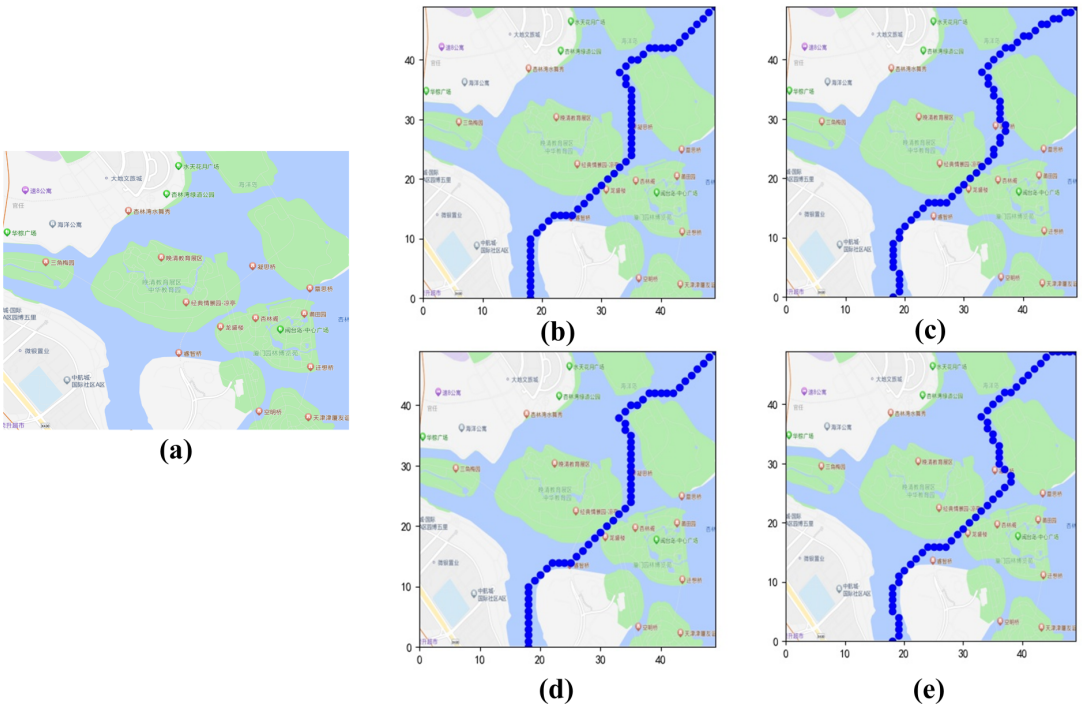

Figure 8. Narrow environment.

To increase the complexity of the map, the narrow water environment with the size of 50×50 is built as shown in Figure 8(a) based on the water around the Garden Expo Park in Xiamen, Fujian Province. The ASV sails from the river below to the upper right corner through this area. The planning paths of each algorithm are shown in Figure 8(b)–(d), and the algorithm order is identical to that of Figure 7.

In Table 4, although the complexity of the environment increases, all four algorithms can plan the path to the final target point. Still, the number of Dijkstra's algorithm searches is larger than the other two algorithms and the A* algorithm is explored 1.5 times more than the enhanced A* algorithm. On

the second search, the number of explorations of the enhanced A* algorithm is only 0. Moreover, the path length is reduced compared to other algorithms, and the algorithmic computation is accelerated by a third or more. In a word, the resources required for path planning are significantly saved.

4.4.3. Extreme conditions

Table 5. Algorithm comparison in extreme conditions.

Algorithm	Environment	Search Times	Path Length	Time
Dijkstra	Maze	197	19.80	37.01
	Narrow	599	80.61	116.0
Enhanced A*	Maze	53	19.00	9.74
	Narrow	145	66.69	26.75

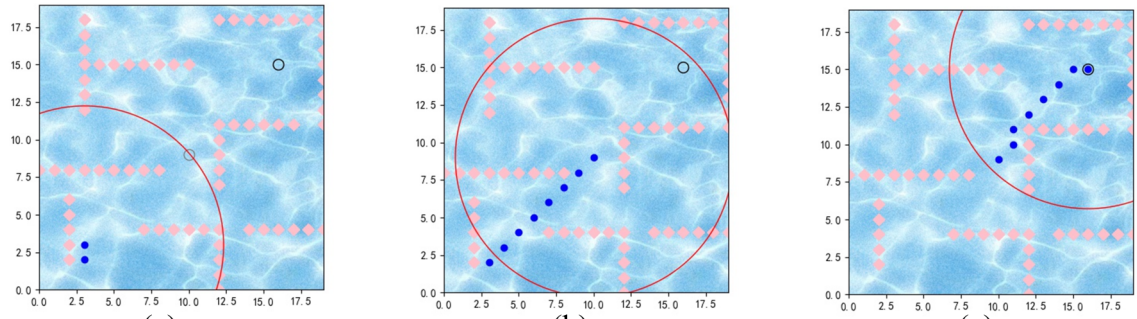

Figure 9. Maze environment with extreme conditions.

The maps are the same as in Figures 8(a) and 7(a), including the start and target points. Due to inclement weather or equipment limitations, only partial information such as the orientation of the target point may be available. The ASV's maximum visibility is represented by the red circle, while the calculated temporary target is gray and the final target point is displayed by the black circle, as shown in Figures 9(a) and 10(a). In such situations, the heuristic function $h(n)$ of A* algorithm and the other A* algorithm in [42] cannot be computed and will degenerate into the Dijkstra algorithm. However, since Dijkstra is a global search algorithm, it will give the same results as in Figures 7(b) and 8(b). As for the enhanced A* algorithm, temporary targets play a crucial role.

By planning the path within the range of visibility, the navigation process is shown in Figures 9 and 10, and the comparison is shown in Table 5. From the starting position as shown in Figure 10(a), the ASV firstly detects the visible area. If there is no final target point, use Eq (3.2) to calculate the temporary target point, and substitute its position into the heuristic function to calculate the priority of nodes in the visible area until the temporary target point is reached. Continue to explore the visible area until the final target point is seen, and the final target point position is substituted into the heuristic function calculation as in other common A* algorithms. The enhanced A* algorithm performs better than Dijkstra algorithm in terms of search times, path length, and algorithm computation time.

In Figure 10(c), the temporary target should have been located on O_1, which is on the line between the target point and the ASV. But its position was covered by an obstacle, in which case the temporary target would be moved laterally in the direction of ASV, that is, exit the obstacle area to O_2.

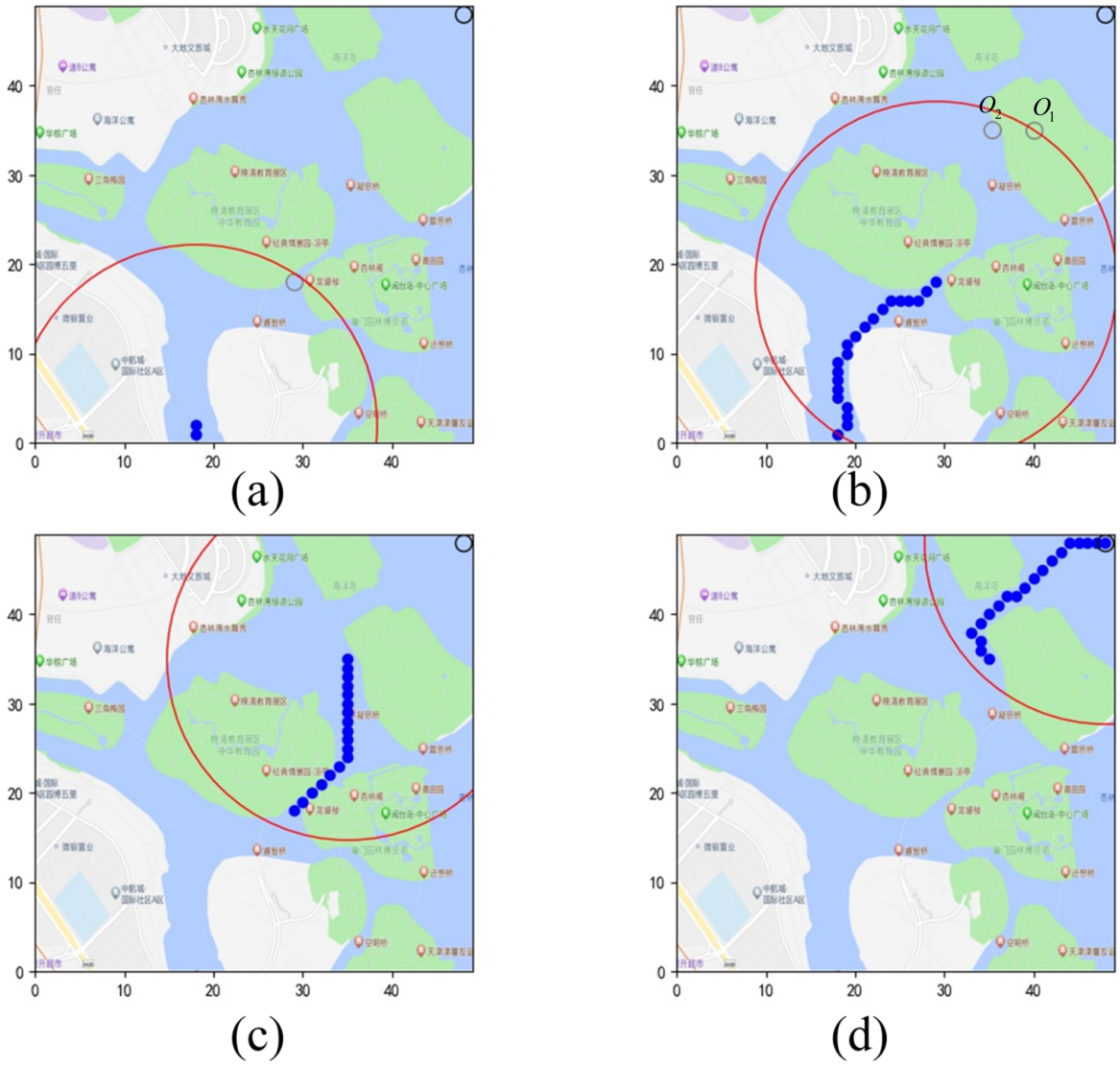

Figure 10. Water area between Xiamen Island and Xiaojinmen Island environment with extreme conditions.

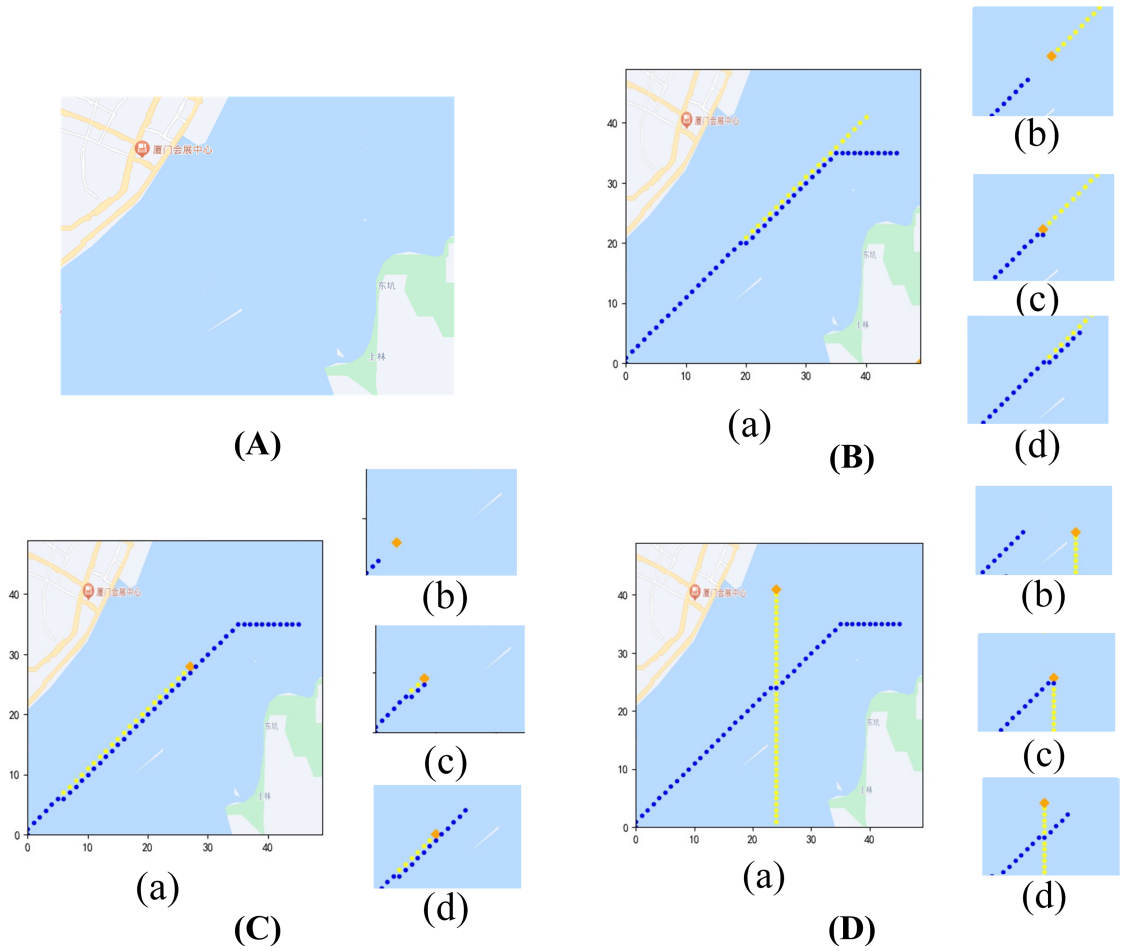

Figure 11. Water area between Xiamen Island and Xiaojinmen Island environment.

4.4.4. Complex water environment

The basic map with the size of 50×50 is shown in Figure 11(A), taking the water area between Xiamen Island and Xiaojinmen Island as an example. The ASV passes through the water from the bottom left to the upper right. A dynamic obstacle is placed, and conducts three conditions of head-on, overtaking and crossing with the ASV, as shown in Figure 11(B),(C). Furthermore, their (a) is the overall path planning diagram, and their (b)–(d) is the obstacle avoidance process.

The proposed algorithm considers the motion mode of ASV and explores the next node from the far right, which is more suitable to the real navigation situation of ASV. In detail, according to Article 14 of the COLREGS, when two ships meet head-on and have the risk of collision, they should take the right turn to avoid the danger. It can be seen in Figure 11(B),(c) that the ASV with the blue track indeed passes over the left side of the dynamic obstacle. Figure 11(C) references Article 13 of the COLREGS. When the ASV intends to overtake the forward dynamic target, it takes a rightward overtaking path. Figure 11(D) references Article 15 of the COLREGS, that is, when an ASV encounters a crossing dynamic obstacle, the ASV with the obstacle on the right is the avoiding side and takes a rightward path adjustment.

5. Conclusions

Aiming at the path planning problems of ASVs in various environments such as a maze, narrow water, complex water, and extreme conditions, we summarizes the three practical situations in the COLREGS. Then the simulation environment is built using the programming language and the validity of the proposed enhanced A* algorithm is verified. The major contributions can be easily summarized in:

1) The enhanced A* method incorporating encrypted memory database for ASV efficient local path planning can deal with three fundamental ASV encounter problems with high performance, including head-on, overtaking, and crossing in the natural water environment.
2) After modifying the heuristic function, the enhanced A* algorithm can effectively accelerate the search efficiency. In particular, compared to Dijkstra's algorithm, the A* algorithm, and the other A* algorithm in [42], the proposed algorithm has a better advantage in the first exploration when the results are similar.
3) Temporary targets can be generated to aid navigation when final target point information is partial. Particularly, in comparison with the Dijkstra algorithm, the A* algorithm and the other A* algorithm in [42], which are degraded to the Dijkstra algorithm since only the approximate direction of the final target point is known. The proposed A* algorithm can achieve advantages in the number of searches, the path length, and the algorithm computation time.
4) In this paper, we consider adding the memory function and saving the compute resource for searching the optimal path when reencountering the same environment. In addition, the memory is encrypted to deal with the transmission leakage problem.

The future research will focus on four major areas: The first goal is to improve the algorithm toward faster computation and safer planning paths. Then, enrich the structure of the proposed algorithm to deal with more complex environments. Furthermore, compress stored map databases to prevent them from requiring too much memory space and taking too much time to search. Moreover, how to apply it in reality, send the path to the real ASV correctly and timely, and control it to navigate according to the path are also the future research areas.

Use of AI tools declaration

The authors declare that they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work is supported by the National Natural Science Foundation of China (52171309, 51809113), Natural Science Foundation of Fujian Province, China (2022J06025, 2021J01843), Fujian Provincial Young Top-Notch Talent Plan (Z02101), and the Research Launch Funding of Jimei University (C623018).

Conflict of interest

The authors declare that there are is no conflicts of interest.

References

1. G. Zhang, X. Shang, J. Liu, W. Zhang, Improved iterative learning path-following control for usv via the potential-based dvs guidance, *Ocean Eng.*, **280** (2023), 114543. https://doi.org/10.1016/j.oceaneng.2023.114543
2. D. Wu, K. Yuan, Y. Huang, Z. M. Yuan, L. Hua, Design and test of an improved active disturbance rejection control system for water sampling unmanned surface vehicle, *Ocean Eng.*, **245** (2022), 110367. https://doi.org/10.1016/j.oceaneng.2021.110367
3. Y. Huang, D. Wu, L. Li, N. Feng, Event-triggered cooperative path following control of multiple underactuated unmanned surface vehicles with complex unknowns and actuator saturation, *Ocean Eng.*, **249** (2022), 110740. https://doi.org/10.1016/j.oceaneng.2022.110740
4. H. Huang, A. V. Savkin, C. Huang, Reliable path planning for drone delivery using a stochastic time-dependent public transportation network, *IEEE Trans. Intell. Transp. Syst.*, **22** (2021), 4941–4950. https://doi.org/10.1109/TITS.2020.2983491
5. N. Wang, Y. Zhang, C. K. Ahn, Q. Xu, Autonomous pilot of unmanned surface vehicles: Bridging path planning and tracking, *IEEE Trans. Veh. Technol.*, **71** (2022), 2358–2374. https://doi.org/10.1109/TVT.2021.3136670
6. N. Feng, D. Wu, H. Yu, A. S. Yamashita, Y. Huang, Predictive compensator based event-triggered model predictive control with nonlinear disturbance observer for unmanned surface vehicle under cyber-attacks, *Ocean Eng.*, **259** (2022), 111868. https://doi.org/10.1016/j.oceaneng.2022.111868
7. N. Wang, C. K. Ahn, Coordinated trajectory-tracking control of a marine aerial-surface heterogeneous system, *IEEE/ASME Trans. Mechatron.*, **26** (2021), 3198–3210. https://doi.org/10.1109/TMECH.2021.3055450
8. Z. You, H. Yan, J. Sun, H. Zhang, Z. Li, Reliable control for flexible spacecraft systems with aperiodic sampling and stochastic actuator failures, *IEEE Trans. Cyber.*, **52** (2022), 3434–3445. https://doi.org/10.1109/TCYB.2020.3008045
9. L. Lei, Y. Zhou, G. Yang, Multisource information fusion-based environment perception and dynamic model of underwater vehicle in irregular ocean environment, *Inf. Fusion*, **94** (2023), 257–271. https://doi.org/10.1016/j.inffus.2023.02.008
10. L. Velasco, J. Balbuena, J. Gonzales, F. Cuella, Development of an asv for oceanographic monitoring on the huarmey coast, in *2021 IEEE 30th International Symposium on Industrial Electronics (ISIE)*, (2021), 1–7. https://doi.org/10.1109/ISIE45552.2021.9576371
11. T. Sato, K. Kim, S. Inaba, T. Matsuda, S. Takashima, A. Oono, et al., Exploring hydrothermal deposits with multiple autonomous underwater vehicles, in *2019 IEEE Underwater Technology (UT)*, (2019), 1–5. https://doi.org/10.1109/UT.2019.8734460

12. C. Specht, E. Świtalski, M. Specht, Application of an autonomous/unmanned survey vessel (asv/usv) in bathymetric measurements, *Polish Marit. Res.*, **24** (2017), 36–44. https://doi.org/10.1515/pomr-2017-0088

13. J. Fraga, J. Sousa, G. Cabrita, P. Coimbra, L. Marques, Squirtle: An ASV for inland water environmental monitoring, in *ROBOT2013: First Iberian Robotics Conference: Advances in Robotics,* (2014), 33–39. https://doi.org/10.1007/978-3-319-03413-3

14. P. Luo, D. Wu, K. Yuan, Y. Yang, Observer-based adaptive integral terminal sliding mode formation control for a vessel train with obstacle avoidance, *Ocean Eng.*, **283** (2023), 115075. https://doi.org/10.1016/j.oceaneng.2023.115075

15. Y. Kawamura, T. Kato, J. Tahara, K. Masakazu, S. Baba, Development μ-ASV using surfboard, in *International Ocean and Polar Engineering Conference*, (2020).

16. H. Wang, S. Diao, Research on application and key technology of usv in target ship, *Ship Electron. Eng.*, **41** (2021), 5–8.

17. T. C. Furfaro, J. E. Dusek, K. D. von Ellenrieder, Design, construction, and initial testing of an autonomous surface vehicle for riverine and coastal reconnaissance, in *OCEANS 2009*, (2009), 1–6. https://doi.org/10.23919/OCEANS.2009.5422207

18. J. Curcio, J. Leonard, A. Patrikalakis, Scout—A low cost autonomous surface platform for research in cooperative autonomy, in *Proceedings of OCEANS 2005 MTS/IEEE*, (2005), 725–729. https://doi.org/10.1109/OCEANS.2005.1639838

19. M. Caccia, Autonomous surface craft: prototypes and basic research issues, in *14th Mediterranean Conference on Control and Automation*, (2006), 1–6. https://doi.org/10.1109/MED.2006.328786

20. L. Li, D. Wu, Y. Huang, Z. M. Yuan, A path planning strategy unified with a colregs collision avoidance function based on deep reinforcement learning and artificial potential field, *Appl. Ocean Res.*, **113** (2021), 102759. https://doi.org/10.1016/j.apor.2021.102759

21. G. Reinelt, *The Traveling Salesman: Computational Solutions for TSP Applications*, Springer, 2003.

22. H. C. Chang, Y. L. Hsu, S. S. Hung, G. R. Ou, J. R. Wu, C. Hsu, Autonomous water quality monitoring and water surface cleaning for unmanned surface vehicle, *Sensors*, **21** (2021). https://doi.org/10.3390/s21041102

23. G. Saiteja, S. Pramod, Design and simulation of an autonomous surface vehicle for trash collection using ros, in *2021 IEEE 9th Region 10 Humanitarian Technology Conference (R10-HTC)*, (2021), 1–6. https://doi.org/10.1109/R10-HTC53172.2021.9641589

24. F. Keppler, N. Dunkelberg, S. Wagner, K. Janschek, Efficient multi-robot path planning for autonomous weed control on complex field configurations, *VDI Ber.*, **2395** (2022), 79–86. https://doi.org/10.51202/9783181023952

25. M. Morin, I. Abi-Zeid, C. G. Quimper, Ant colony optimization for path planning in search and rescue operations, *Eur. J. Oper. Res.*, **305** (2023), 53–63. https://doi.org/10.1016/j.ejor.2022.06.019

26. C. Lu, J. Yang, Path planning of subsea mining vehicle, in *Proceedings of the International Offshore and Polar Engineering Conference*, (2021), 368–374.

27. A. K. Sadhu, A. Konar, T. Bhattacharjee, S. Das, Synergism of firefly algorithm and q-learning for robot arm path planning, *Swarm Evol. Comput.*, **43** (2018), 50–68. https://doi.org/10.1016/j.swevo.2018.03.014

28. S. Islam, A. Razi, A path planning algorithm for collective monitoring using autonomous drones, in *2019 53rd Annual Conference on Information Sciences and Systems (CISS)*, (2019), 1–6. https://doi.org/10.1109/CISS.2019.8693023

29. W. J. Harlin, D. A. Cicci, Ballistic missile trajectory prediction using a state transition matrix, *Appl. Math. Comput.*, **188** (2007), 1832–1847. https://doi.org/10.1016/j.amc.2006.11.048

30. J. Hu, D. Yan, J. Zheng, Embed behavior decision making into ship collision avoidance path planning based on ant colony and q-learning algorithm, *Ind. Eng. Innovation Manage.*, **5** (2022), 20–28. https://doi.org/10.23977/ieim.2022.050105

31. L. Yu, T. Qian, X. Ye, F. Zhou, Z. Luo, A. Zou, Research on obstacle avoidance strategy of usv based on improved grid method, in *4th International Symposium on Power Electronics and Control Engineering (ISPECE 2021)*, 2021. https://doi.org/10.1117/12.2620444

32. D. Xue, D. Wu, A. S. Yamashita, Z. Li, Proximal policy optimization with reciprocal velocity obstacle based collision avoidance path planning for multi-unmanned surface vehicles, *Ocean Eng.*, **273** (2023), 114005. https://doi.org/10.1016/j.oceaneng.2023.114005

33. N. Wang, H. He, Y. Hou, B. Han, Model-free visual servo swarming of manned-unmanned surface vehicles with visibility maintenance and collision avoidance, *IEEE Trans. Intell. Transp. Syst.*, **2023** (2023), 1–13. https://doi.org/10.1109/TITS.2023.3310430

34. J. Yu, Z. Chen, Z. Zhao, P. Yao, J. Xu, A traversal multi-target path planning method for multi-unmanned surface vessels in space-varying ocean current, *Ocean Eng.*, **278** (2023), 114423. https://doi.org/10.1016/j.oceaneng.2023.114423

35. J. B. Escario, J. F. Jimenez, J. M. Giron-Sierra, Optimisation of autonomous ship manoeuvres applying ant colony optimisation metaheuristic, *Exp. Syst. Appl.*, **39** (2012), 10120–10139. https://doi.org/10.1016/j.eswa.2012.02.069

36. Y. Singh, S. Sharma, R. Sutton, D. Hatton, Path planning of an autonomous surface vehicle based on artificial potential fields in a real time marine environment, in *COMPIT'17: 16th International Conference on Computer and IT Applications in the Maritime Industries*, (2017), 48–54. https://doi.org/10.1016/j.automatica.2004.10.006

37. R. Skjetne, T. I. Fossen, P. V. Kokotović, Adaptive maneuvering, with experiments, for a model ship in a marine control laboratory, *Automatica*, **41** (2005), 289–298. https://doi.org/10.1016/j.automatica.2004.10.006

38. E. Demirel, D. Bayer, Further studies on the COLREGs (collision regulations), *TransNav*, **9** (2015), 17–22. https://doi.org/10.12716/1001.09.01.02

39. E. W. Dijkstra, A note on two problems in connexion with graphs, in *Edsger Wybe Dijkstra: His Life,Work, and Legacy*, Association for Computing Machinery, (2022), 287–290. https://doi.org/10.1145/3544585

40. P. E. Hart, N. J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, *IEEE Trans. Syst. Sci. Cyber.*, **4** (1968), 100–107. https://doi.org/10.1109/TSSC.1968.300136

41. H. Lyu, Y. Yin, Colregs-constrained real-time path planning for autonomous ships using modified artificial potential fields, *J. Navig.*, **72** (2019), 588–608. https://doi.org/10.1017/S0373463318000796

42. Z. Wang, X. Xiang, Improved astar algorithm for path planning of marine robot, in *2018 37th Chinese Control Conference (CCC)*, (2018), 5410–5414. https://doi.org/10.23919/ChiCC.2018.8483946

AIMS Press