Automation and Machine Learning (2026)
Clausius Scientific Press, Canada

DOI: 10.23977/autml.2026.070102
ISSN 2516-5003 Vol. 7 Num. 1

Design and Application of an A–Fuzzy Path Planning Algorithm for Unmanned Surface Vehicle*

Tianxiang Yang[1,a,*], Liping Wen[2,b], Yiming Jia[1,c]

[1] *School of Ocean Engineering, Jiangsu Ocean University, Lianyungang, 222005, China*
[2] *School of Innovation and Entrepreneurship, Jiangsu Ocean University, Lianyungang, 222005, China*
[a] *yangtx_7@126.com*, [b] *2239252345@qq.com*, [c] *jia_ym@yeah.net*
**corresponding author*

Keywords: Improved A* algorithm, T-S fuzzy inference, Map binary segmentation, Global path planning method, Safety margin

Abstract: This paper presents a global path planning algorithm for unmanned surface vehicle (USV) based on an A*–fuzzy hybrid approach that integrates fuzzy collision-avoidance reasoning. To address path planning challenges in real-world scenarios, the proposed method first processes actual satellite imagery through image preprocessing and binarization to generate a grid-based navigable map. The grid map is then scaled to physical dimensions, and obstacle boundaries are dilated according to the USV minimum safety radius to ensure navigational clearance. Subsequently, the A* algorithm is employed for initial path search, while a Takagi-Sugeno (T-S) fuzzy inference model is applied to refine node selection near obstacles, enhancing local decision-making under uncertainty. Finally, the generated trajectory is simplified by retaining only critical waypoints, significantly reducing data storage requirements without compromising path quality. Simulation results demonstrate that the proposed algorithm improves both redundancy and safety in USV navigation, maintains high computational efficiency, and offers a practical solution for autonomous maritime decision-making. The approach effectively balances path optimality, obstacle avoidance capability, and memory efficiency, providing valuable support for safe USV operations.

1. Introduction

As unmanned surface vehicle (USV) operating on water, unmanned boats offer advantages such as design flexibility, substantial payload capacity, and high efficiency compared to aerial drones or land-based unmanned vehicles. Having experienced vigorous development in recent years, they are now widely deployed in scientific research, surveying, mapping, and military applications. Research into path planning tailored for unmanned boats can enhance their autonomous navigation capabilities and intelligence, holding significant practical importance.

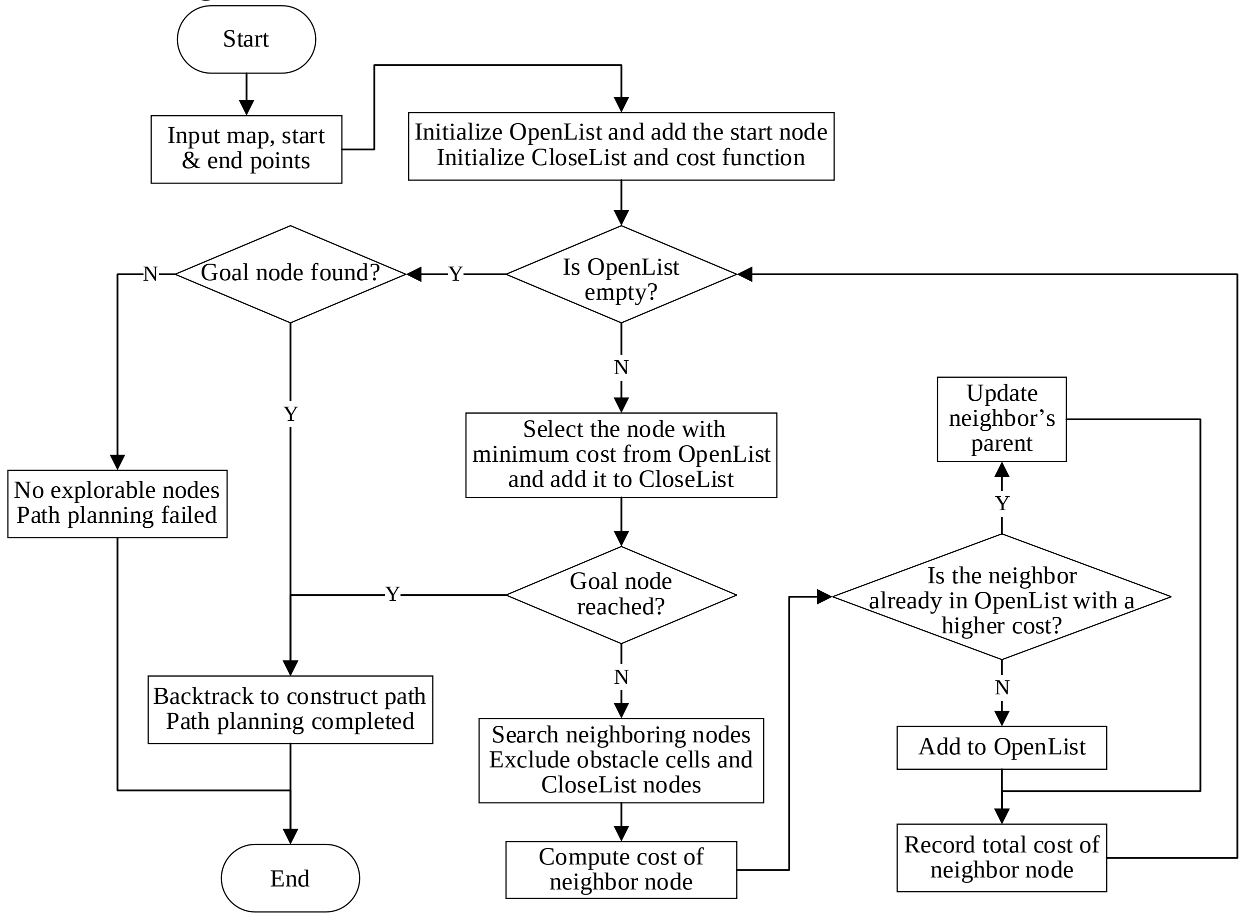
Currently, prevalent path planning approaches can be categorised as follows: search/sampling-based algorithms, such as Dijkstra, A*, and RRT; optimisation algorithms mimicking natural behavioural patterns, including ant colony optimisation, genetic algorithms, particle swarm optimisation, and grey wolf algorithms; field algorithms inspired by physical spaces, such as artificial potential field (APF) methods; and action strategy planning algorithms utilising AI exploration learning, such as Q-Learning based on reinforcement learning. Among these, the heuristic search algorithm A* has gained widespread application due to its integration of the respective advantages of Dijkstra and greedy algorithms, coupled with its simplicity of implementation and high search efficiency.

In recent years, various improved forms of path planning algorithms based on A* for unmanned surface vehicle (USV) have been proposed. Ehsan Adel Rastkhiz et al. [1] employed fuzzy set theory to construct planned path points, static/dynamic obstacles, and environmental uncertainties, generating fuzzy paths for autonomous navigation. However, this path-based approach struggles to maintain real-time performance amid increasing environmental complexity while demanding greater computational resources; Yang Xu et al. [2] proposed an algorithm integrating enhanced A* with fuzzy control DWA for unmanned vehicle path planning. Fuzzy rules were applied to adjust weights in DWA's evaluation function concerning heading angle, obstacle avoidance, velocity, and correction distance; Charis Ntakolia et al. [3] incorporated fuzzy decision logic into the Swarm Intelligence Graph-based Pathfinding Algorithm (SIGPA). They employed both Mamdani and Takagi-Sugeno–Kang models to optimise search path quality, adapting the algorithm for USV. Gregorius Airlangga et al. [4] proposed a knowledge-driven fuzzy enhanced A* algorithm for multi-agent systems. Adjustment factors derived from obstacle distances and path angles input to the fuzzy system dynamically modulate cost function and penalty function coefficients, enabling the algorithm to account for obstacles and diverse path attributes during search; V. Sangeetha et al. [5] proposed a fuzzy reasoning A* algorithm for unknown environments, employing a Mamdani model. They designed a fuzzy system with six inputs and one output, considering steering, angle, forward distance, distance to target point, left diagonal distance, and right diagonal distance. However, the experimental simulation environment was relatively simplistic. Furthermore, the design comprised only 25 fuzzy inference rules, lacking validation of search efficiency in complex terrain. Accordingly, most path planning studies for unmanned vessels focus on combining A* with artificial potential field methods or DWA. A* algorithms enhanced with fuzzy reasoning predominantly employ Mamdani models, which exhibit lengthy execution times and suboptimal inference efficiency in complex terrain. Furthermore, the selection of planning terrain or processing of electronic charts tends towards idealised scenarios, lacking design validation grounded in real-world operational contexts for USV.

This paper designs improvements based on the A* global planning algorithm, integrating fuzzy reasoning for obstacle avoidance and incorporating human natural language experience as a pathfinding reference. It fully considers the USV's self-influencing factors during operation to adapt it to practical engineering applications. Firstly, actual satellite imagery is selected for image preprocessing and binarization to generate a raster map suitable for path search; Secondly, the raster map undergoes scale conversion to determine the USV minimum safety radius, enabling boundary expansion processing for map obstacles. Subsequently, based on the A* algorithm's 8-neighbourhood search strategy, obstacles are processed coherently. The T-S (Takagi-Sugeno) model is employed for fuzzy inference on trajectory points near obstacles, accelerating inference speed. Finally, the planned trajectory undergoes simplification, retaining only critical path point information. Simulations demonstrate that the proposed algorithm enhances the redundancy and safety of USV path planning, proving practical while significantly reducing path point storage requirements without compromising search efficiency.

2. Introduction to the A* Algorithm

The traditional A* algorithm is a graph-based heuristic search method, primarily comprising a grid map, a starting point, a destination endpoint, a list of marked path nodes (Closelist), a list of unexplored neighbouring nodes (Openlist), and a cost function. Its implementation steps are illustrated in Figure 1.



Start
Input map, start & end points
Initialize OpenList and add the start node Initialize CloseList and cost function
Is OpenList empty?
Goal node found?
N
Y
No explorable nodes Path planning failed
Select the node with minimum cost from OpenList and add it to CloseList
Update neighbor's parent
Goal node reached?
Is the neighbor already in OpenList with a higher cost?
Backtrack to construct path Path planning completed
Search neighboring nodes Exclude obstacle cells and CloseList nodes
Add to OpenList
End
Compute cost of neighbor node
Record total cost of neighbor node

Figure 1: Flowchart of A* path search algorithm.

The Closelist stores path point information for the final trajectory to be planned and constructed, while the Openlist holds details of all currently explored neighbouring nodes. At each step, the node with the optimal cost is selected from the Openlist and transferred to the Closelist, forming the candidate node set for the Closelist. In each iterative step, the A* algorithm's search is based on selecting the node with the optimal cost, defined by the following cost calculation function:

$$f(n) = g(n) + h(n) \tag{1}$$

where: $f(n)$ denotes the total cost value of the n-th node; $g(n)$ denotes the accumulated path cost from the start node to the current node n, computed as the sum of the traveled distances across the traversed grid cells; $h(n)$ is the estimated cost distance from the node n to the endpoint. Estimated cost distance calculations typically employ Manhattan distance, Euclidean distance, etc. This paper uses Euclidean distance to compute the estimated cost distance, defined as:

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

where x_n and y_n denote the coordinates of node n on the grid map, x_{goal} and y_{goal} denote the coordinates of the goal (target) endpoint.

3. Image Preprocessing

This section constructs a raster map by selecting a Google satellite image of the Jing Si Lake area within Jiangsu Ocean University's Cangwu Campus, with a pixel size of 1100×1000px. The colour satellite image is processed to convert it into a binary raster map, preparing for subsequent algorithm design tasks such as pathfinding and route planning. The process of converting the satellite map into a binary raster map is illustrated in Figure 2.

The design process for constructing a raster map is as follows:

(1) Firstly, perform K-means clustering analysis on the extracted satellite imagery. K-means clustering is an unsupervised machine learning classification method. Figure 2(a) primarily contains three distinct colour elements: vegetation, water bodies, and roads. Therefore, the objective was set to partition the dataset into three clusters. Through iterative updates using the K-means algorithm, the image segmentation effect shown in Figure 2(b) was achieved.

(2) Apply a closing operation to the image in Figure 2(b). The closing operation involves first dilating the image, followed by erosion. This process effectively fills gaps within the colour blocks delineated in Figure 2(b), while connecting adjacent objects to achieve smoother boundaries, as demonstrated in Figure 2(c). A square convolution kernel of size 8×8 pixels was employed for the closing operation.

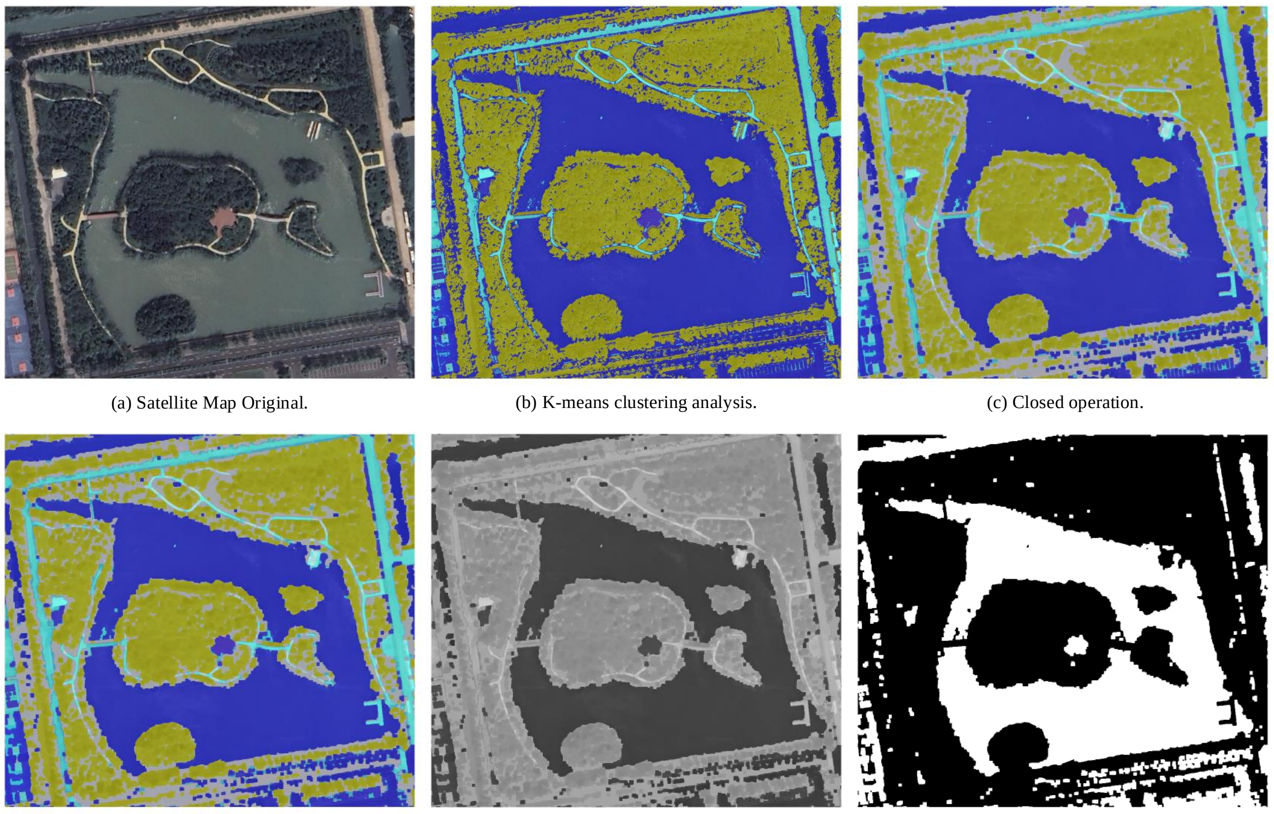

(a) Satellite Map Original.
(b) K-means clustering analysis.
(c) Closed operation.
(d) Opening operations
(e) Grayscale image.
(f) Binarized raster map.

Figure 2: Construction of a grid map from a satellite map.

(3) Subsequently, apply an opening operation to smooth and denoise the colour blocks in Figure 2(c). This involves first eroding the image followed by dilation, which removes boundary artefacts

while eliminating minute objects to achieve noise reduction, as shown in Figure 2(d). The opening operation employs a 3×3 pixel square convolution kernel.

(4) Following morphological processing, the resulting three-colour image in Figure 2(d) is converted to greyscale, as shown in Figure 2(e).

(5) Finally, the Otsu thresholding method is employed to adaptively determine the binarization threshold for the greyscale image, resulting in a black-and-white binary image as depicted in Figure 2(f). Within this image, black blocks represent obstacles, while white blocks denote areas suitable for path planning. This yields the final raster map for path planning, sized 1100×1000 pixels, consistent with the original satellite image.

4. A*-Fuzzy Algorithm Design Process

4.1. Map Grid Scale Definition

Prior to defining the safety radius during USV operation, the map must undergo scale conversion to normalise the actual distance corresponding to each pixel point in the grid map. Employing the built-in scale tool within Google Satellite Maps, measurements of the selected area's dimensions reveal that 1100×1000 px corresponds to actual dimensions of 303×277.5 m. This equates to each horizontal pixel representing 0.2755 m/px, and 0.2775 m/px vertically. To simplify the problem while ensuring safety thresholds, the minimum value is adopted as the baseline, treating each pixel as representing 0.2755 m/px in both horizontal and vertical directions.

4.2. Defining Safety Distances

During global path planning, prior information should be maximally utilised to mitigate collision risks between the vessel and static obstacles, thereby introducing redundancy for actual waterway navigation. The algorithm design accounts for intrinsic USV characteristics, as illustrated in Figure 3, by determining the vessel's minimum safety radius as a reference for terrain obstacle expansion. Grid points within this expanded zone are prohibited from inclusion in the planned route.

The minimum safety radius of the USV, denoted by R_s, is defined as:

$$R_s = R_b + R_d + R_m \tag{3}$$

where R_b is the semi-length of the vessel's longest side, i.e., the radius of the vessel's circumscribed circle, taken as $R_b = 0.43\,\text{m}$ in this study; R_d is the braking distance of the vessel at operational speed, typically obtained experimentally, taken as $R_d = 0.3\,\text{m}$ here; R_m is the fluctuation radius of the positioning error from the vessel's GNSS module under normal ideal operating conditions, to be statistically analysed through specific experiments below.

Typically, satellite positioning modules installed on actual vessels are categorised into two types: RTK positioning and GNSS single-point positioning. RTK positioning offers higher accuracy and lower error margins but substantially increases overall costs. It also requires coordination with base station positioning during operation, limiting its application scenarios. Consequently, it is predominantly employed on survey vessels or unmanned vessels/boats demanding high control precision. GNSS single-point positioning is widely adopted due to its low cost, requiring only a single module and antenna for operation. It offers flexible deployment scenarios but achieves lower positioning accuracy than RTK methods, with traditional single-point positioning typically achieving metre-level precision. This study investigates low-cost unmanned vessel applications using a standard GNSS single-point positioning module, specifically the ATGM336H model from

Zhongke Microelectronics, commercially available at a price of merely 60 yuan. The following presents statistical analysis of experimental data collected under open-sky conditions and ideal operating states for this positioning module.

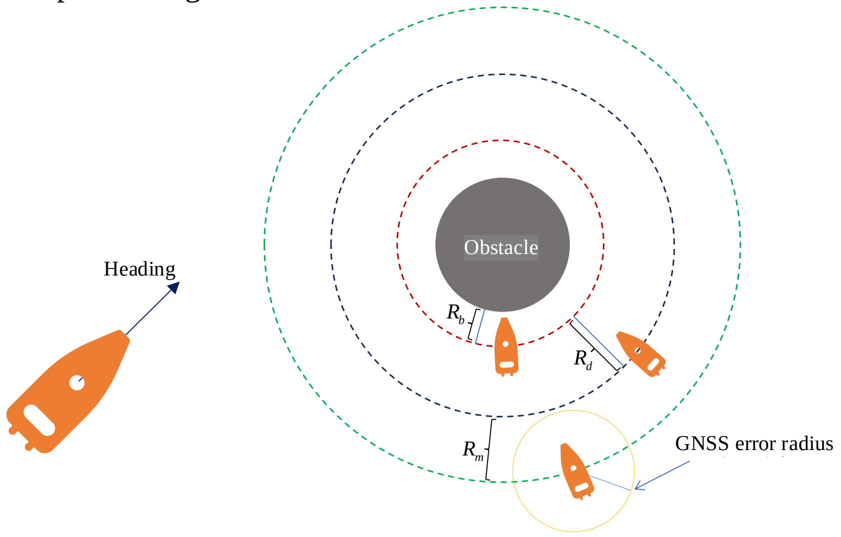


Figure 3: Safety-distance zoning for the unmanned surface vehicle (USV).

The experiment employed the antenna supplied with the GNSS module, which featured a typical nominal gain of 38 dB. It was secured using a tripod and 3D-printed structure, mounted at a height of 1.7 metres above ground level, as illustrated in Figure 4.

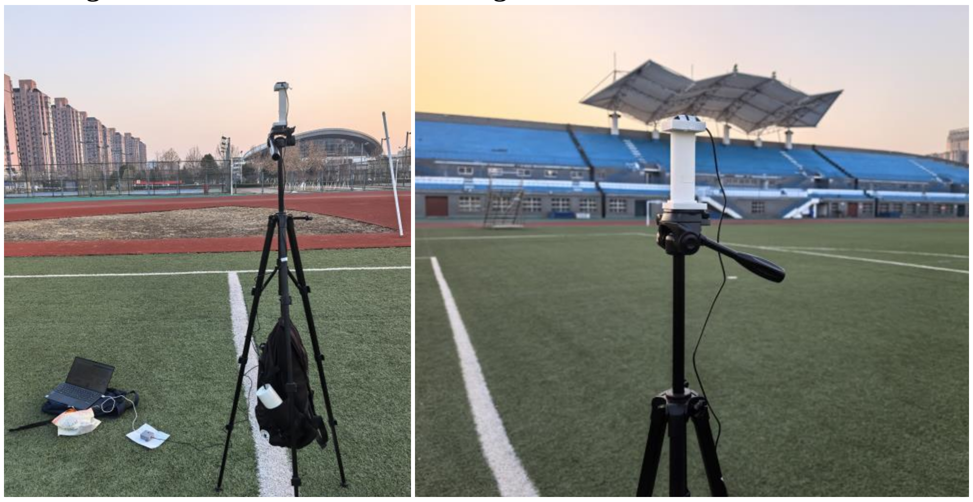


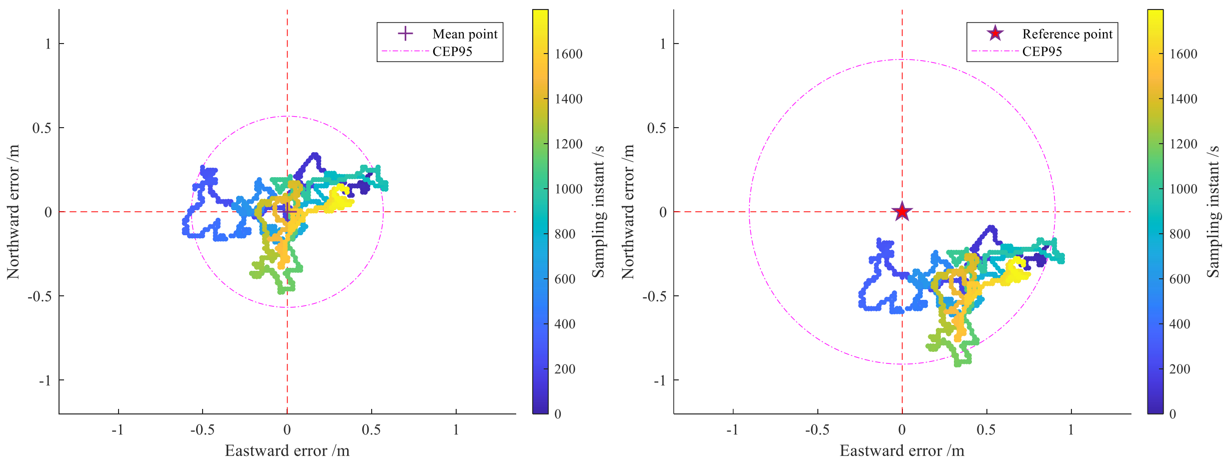
Figure 4: GNSS module used in the positioning experiment setup.

The GNSS accuracy evaluation metrics employed [6] primarily encompass: Root Mean Square Error (RMSE), 95% Circular Error Probable (CEP95), and standard deviation of errors.

Due to experimental constraints, including the absence of high-precision RTK positioning equipment for acquiring absolute coordinates of test points, this study utilised coordinates of prominent landmarks on Google Maps [7-9] (e.g., intersections of football pitch boundary lines, synthetic running track markings) as reference benchmarks. Concurrently, the GNSS module was positioned at these actual locations. Measured data were then compared against benchmark coordinates to derive the absolute error of the module's static positioning.

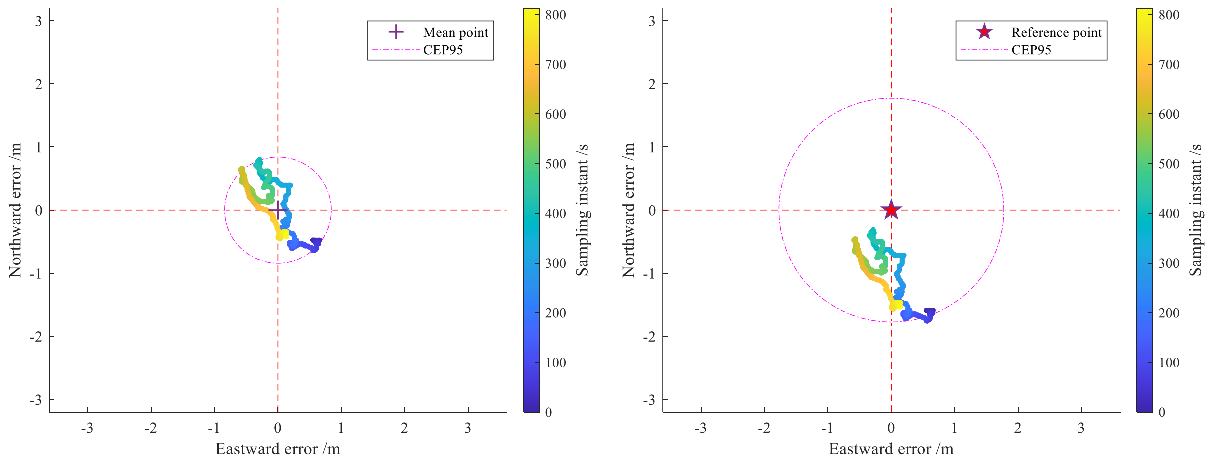
(1) Static positioning tests. As illustrated in Figures 5 to 7, the module underwent operational testing across diverse scenarios and varying positioning frequencies. The resulting static latitude-

longitude data were compared against both the overall mean point and Google Maps reference points. This yielded two-dimensional scatter plots of relative and absolute errors, colour-coded according to sampling time.



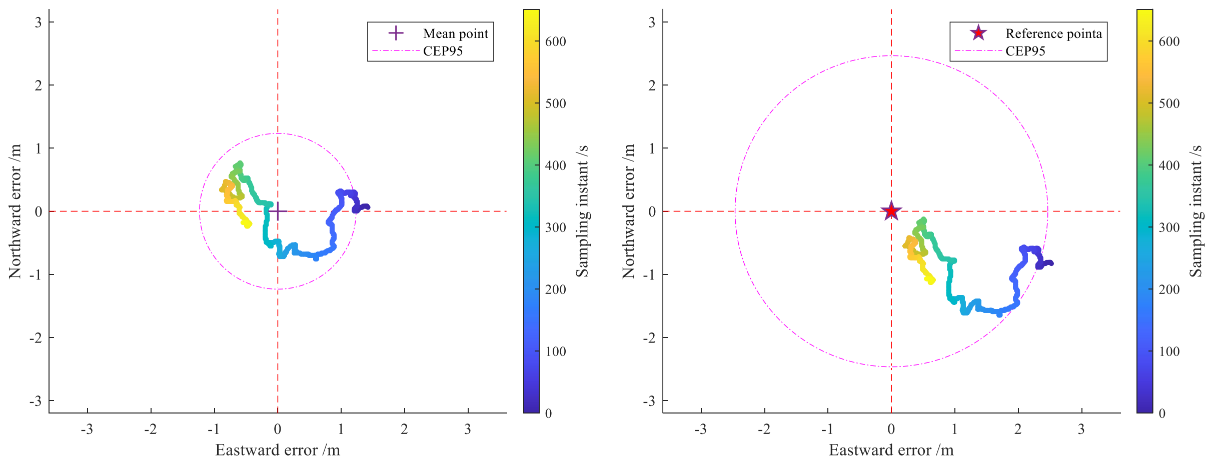
(a) Relative error (b) Absolute error

Figure 5: Planar positioning errors at 1 Hz in an open land area.



(a) Relative error (b) Absolute error

Figure 6: Planar positioning errors at 1 Hz near the boundary of an open water area.



(a) Relative error (b) Absolute error

Figure 7: Planar positioning errors at 2 Hz near the boundary of an open water area.

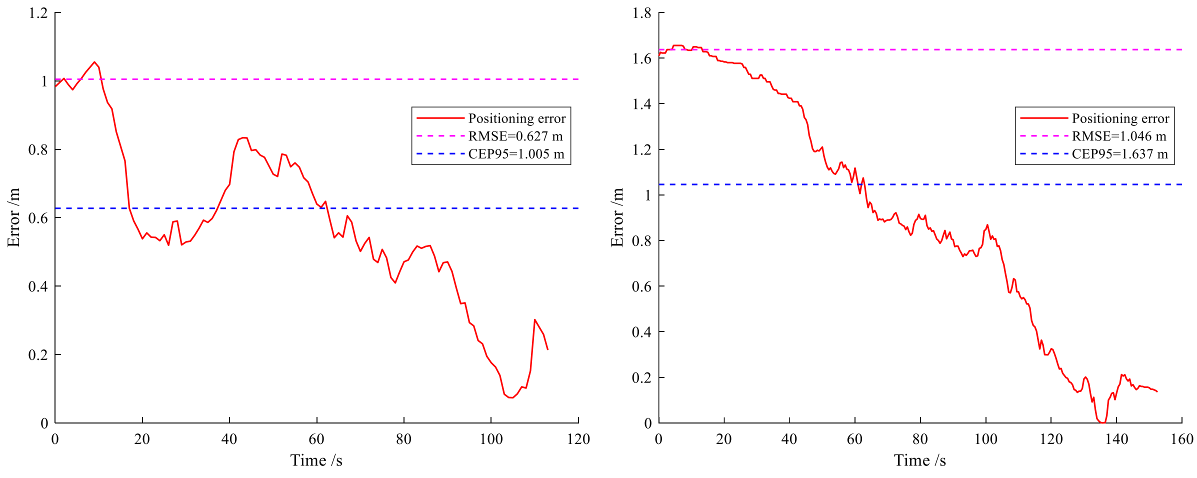
The test results of the GNSS module are summarized in Table 1. Under the same positioning update rate across different scenarios, the positioning error in the open-land setup is smaller than that near the boundary of an open water area. This is mainly attributed to multipath effects caused by water-surface reflections, as well as the module's oscillation induced by lake-surface ripples when it is mounted on a floating buoy.

Table 1: Analysis of static positioning test data of the GNSS module.

Test data analysis		Open terrain 1Hz static positioning	Static positioning at 1Hz near open water	Static positioning at 2Hz near open water
Google Maps coordinates		34.61122600 °N 119.21736000 °E	34.61048828 °N 119.21568031 °E	34.61048828 °N 119.21568031 °E
Mean coordinates from GNSS module positioning		34.61122212 °N 119.21736397 °E	34.61047828 °N 119.21568031 °E	34.61048028 °N 119.21569232 °E
Relative mean error	RMSE (m)	0.32	0.55	0.82
	CEP95 (m)	0.57	0.84	1.23
Absolute error	RMSE (m)	0.65	1.24	1.64
	CEP95 (m)	0.90	1.77	2.47

(2) Linear dynamic positioning test. A 100-metre section of synthetic track on an open athletics field was selected. A GNSS module was manually held while maintaining relatively uniform linear motion. Google Maps was used to connect the start and end coordinates of this 100-metre straight line as the baseline. Statistical error analysis was performed on the measured module data, as shown in Figure 8.

Considering practical requirements, the GNSS modules employed were subjected to performance testing at different positioning frequencies, with results presented in Table 2. The results indicate that, under the same influencing conditions (e.g., human-induced effects during motion and coordinate errors in Google Maps), all error metrics at 1 Hz outperform those at 2 Hz.



(a) 1 Hz GNSS positioning update rate (b) 2 Hz GNSS positioning update rate

Figure 8: Error curves in the straight-line dynamic test.

Table 2: Analysis of dynamic positioning test data of the GNSS module.

Positioning frequency	RMSE (m)	CEP95 (m)	Maximum error (m)	Standard deviation of the error (m)
1Hz	0.627	1.005	1.055	0.247
2Hz	1.046	1.637	1.655	0.526

Based on the measured data, considering the actual operating environment of the GNSS module and the positioning acquisition frequency, the CEP95 test metric for 1Hz static positioning at the edge of open water is taken as the benchmark. The magnitude of the error term $R_m = 1.77\,\text{m}$ is selected, representing the minimum safe radius $R_s = 0.43 + 0.3 + 1.77 = 2.5\,\text{m}$ for the USV.

Based on the aforementioned prior information, the safety distance delineation process is undertaken. Terrain boundaries undergo expansion processing, which involves imposing hard constraints to establish prohibited zones, thereby preventing collisions between USV and static obstacles. The pixel size of the expanded terrain boundary is determined by the safety distance L_{safe}. To account for safety and redundancy, it should satisfy $L_{safe} \geq R_s$, here taken as $L_{safe} = 2.5\,\text{m}$, The expanded grid size $L_c = \dfrac{L_{safe}}{0.2755\ \text{m/px}} \approx 9\ \text{px}$ of the terrain boundary can be calculated accordingly. The terrain after expansion is shown in Figure 9.

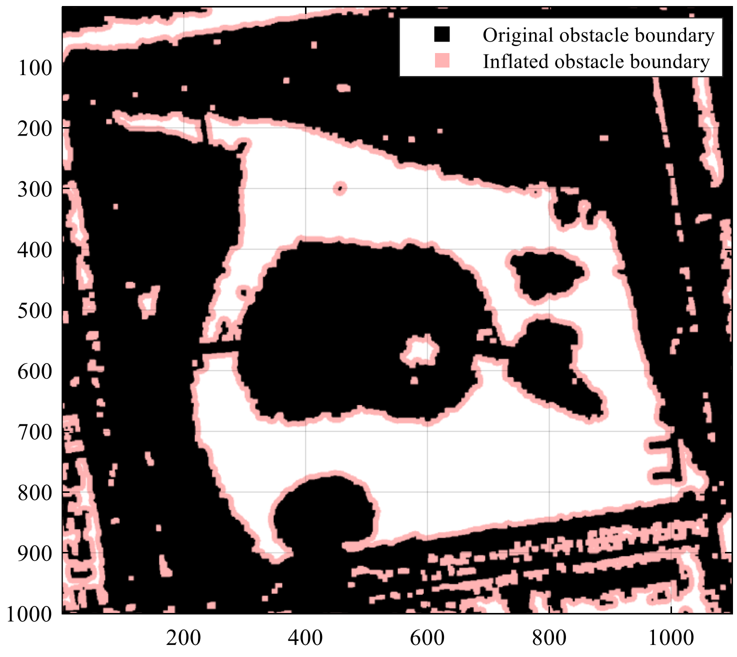


Figure 9: Terrain inflation (inflation radius: 9 grid cells).

4.3. A* Search Strategy Based on Fuzzy Reasoning Using the T-S Model

This paper employs an A* algorithm with an 8-neighbourhood search strategy, which accelerates search time while maintaining flexible directionality, as illustrated in Figure 10. The horizontal and vertical movement distances for each grid cell are defined as 1, while movement along a 45-degree diagonal direction is defined as $\sqrt{2}$. The heuristic function utilises Euclidean distance. Furthermore, handling of contiguous obstacle scenarios is implemented by prohibiting traversal of diagonal obstacle regions during search operations, as depicted in Figure 11.

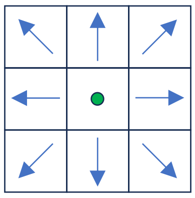


Figure 10: Eight-neighborhood search directions.

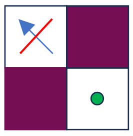


Figure 11: Forbidden search region caused by diagonal obstacles.

This paper designs the cost calculation function for each node in the path planning algorithm as follows:

$$f(n) = g(n) + h(n) + \beta \cdot y(n) \tag{4}$$

In the equation, $y(n)$ denotes the distance cost of obstacle n in the fuzzy inference grid map, while β represents the adjustment coefficient for this cost.

Previous studies predominantly employed the Mamdani model for fuzzy inference in pathfinding algorithm optimisation and collision avoidance. This paper adopts the T-S model [10] to refine the A* algorithm's planning path and perform fuzzy inference on obstacle distance costs. This approach enhances path search efficiency while ensuring the USV safe navigation.

The design philosophy of the T-S fuzzy model involves achieving the fitting of an entire nonlinear system by weighting and combining multiple linear systems through semantic fuzzification. Its form is expressed as:

$$R^i : \text{IF } f(x_1 \text{ is } A_1^i, x_2 \text{ is } A_2^i, \cdots, x_k \text{ is } A_k^i) \quad \text{THEN } y_i = g(x_1, \cdots, x_k) \tag{5}$$

In Equation (5), R^i denotes the i-th rule of the fuzzy system, y_i represents the output value of the i-th rule, x_k is the k-th input state variable, and A_k^i is the fuzzy set of the i-th input state variable in the k-th rule, with its membership function expressed as $\mu_{A_k^i}(x_k)$, $f(\cdot)$ denotes the relationship between the input state variable and its corresponding fuzzy set, while $g(\cdot)$ represents the linear or non-linear function of this rule, $i = 1, 2, \cdots, r$ 。

The final output y of the fuzzy system is the weighted average of all rule outputs, generally expressed as:

$$y = \frac{\sum_{i=1}^{r} \omega_i y_i}{\sum_{i=1}^{r} \omega_i}, \ \omega_i = \prod_{n=1}^{k} \mu_{A_n^i}(x_n) \tag{6}$$

In Eq. (6), ω_i denotes the membership value of the fuzzy set under Rule i, where $\mu_{A_n^i}$ represents the membership function of fuzzy set A_k^i.

Let state variable d denote the distance from the obstacle boundary. This paper designs three rules for fuzzy inference regarding the distance cost n of node $y(n)$ to obstacles in the grid map. A fuzzy set is formulated to express the membership relationship of state variable d, categorised as Near, Mid, and Far. The linear outputs corresponding to each rule are shown in Equation 7:

$$\begin{aligned} &R^1 : \text{IF } d \text{ is Near} \quad \text{THEN } y_1 = 1.0 \\ &R^2 : \text{IF } d \text{ is Mid} \quad \text{THEN } y_2 = 0.5 \\ &R^3 : \text{IF } d \text{ is Far} \quad \text{THEN } y_3 = 0.0 \end{aligned} \tag{7}$$

The membership function for the fuzzy set is selected as illustrated in Figure 12.

Ultimately, the output formula for the node's distance-cost fuzzy inference regarding obstacles is as follows:

$$y(n) = \frac{\omega_1 y_1 + \omega_2 y_2 + \omega_3 y_3}{\omega_1 + \omega_2 + \omega_3} \tag{8}$$

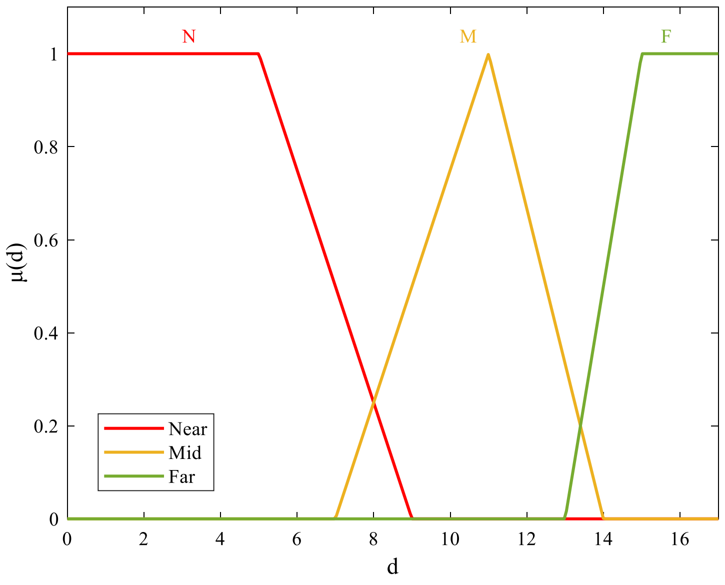


Figure 12: Membership functions for fuzzy inference.

4.4. Page Setup

For the set of path points successfully mapped by A*–fuzzy search, practical implementation constraints imposed by embedded processors such as MCUs necessitate path simplification. This involves retaining key inflection points with high information density from the planned path, connecting points via straight lines to preserve the overall shape of the original path, thereby reducing data storage requirements and computational load. The Ramer-doulas-Peucker (RDP) algorithm [11] constitutes an iterative fitting method for path curve thinning and simplification. This study employs the RDP algorithm to reduce path curve complexity whilst ensuring navigational safety of the simplified path points.

Let the set of path points after planning be denoted as $C=(P_1,P_2,P_3,\cdots,P_n)$. The implementation principle of the RDP algorithm is as follows: Define a distance threshold ε, Select starting point P_1 and endpoint P_n to establish a straight line connection P_1P_n, Examine sequentially the projection distance between each point P_i in the path point set C and the line segment. When the distance of any point P_m exceeds the threshold ε, designate that point as a connection point. Remove segment P_1P_n, re-establish connecting segments P_1P_m and P_mP_n, and partition point set C into two path subsets, $C_1=(P_1,P_2,P_3,\cdots,P_m)$ and $C_2=(P_m,P_{m+1},P_{m+2},\cdots,P_n)$. In C_1 and C_2 respectively, examine the distances between $P_k,\ k\in[1,2,\cdots,m]$ and P_1P_n, and between $P_j,\ j\in[m,m+1,\cdots,n]$ and P_mP_n, determining whether they exceed the threshold ε. Establish new connection relationships accordingly, iterating this process until all distances from points to line segments are less than or equal to the threshold. This yields the final simplified set of path points C_s.

In summary, the overall design steps and execution flow of the A*–fuzzy algorithm are illustrated in Figure 13.

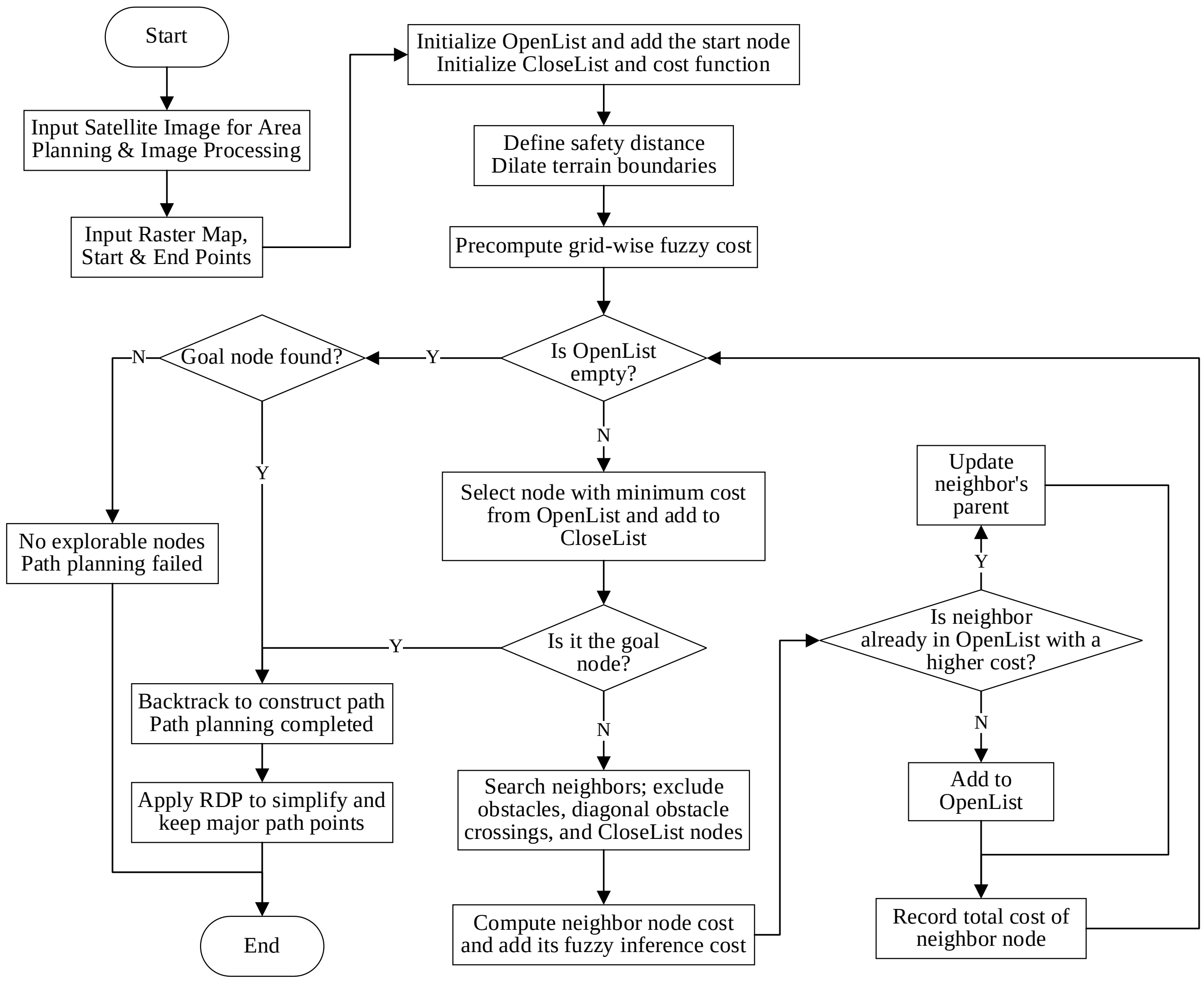

Start
Input Satellite Image for Area Planning & Image Processing
Input Raster Map, Start & End Points
Initialize OpenList and add the start node
Initialize CloseList and cost function
Define safety distance
Dilate terrain boundaries
Precompute grid-wise fuzzy cost
Is OpenList empty?
Y
Goal node found?
N
No explorable nodes
Path planning failed
Y
N
Select node with minimum cost from OpenList and add to CloseList
Is it the goal node?
Y
Backtrack to construct path
Path planning completed
Apply RDP to simplify and keep major path points
End
N
Search neighbors; exclude obstacles, diagonal obstacle crossings, and CloseList nodes
Compute neighbor node cost and add its fuzzy inference cost
Update neighbor's parent
Y
Is neighbor already in OpenList with a higher cost?
N
Add to OpenList
Record total cost of neighbor node

Figure 13: Overall flowchart of the proposed A*–fuzzy path planning method.

5. Simulation Experiment Comparison

The processor model employed by the algorithm verification platform is the i9-13900HX, with 64GB of memory and a Windows 11 operating system environment. The comparative results of different algorithms are illustrated in Figure 14, where the blue dots denote the starting points, the red hexagrams represent the endpoints, and the planned paths are depicted by green lines. Figure 14(a) presents the path search results of the traditional A* algorithm. Without safety buffer constraints, it prioritises directions closer to the endpoint, resulting in a path hugging obstacle edges with considerable complexity. This poses navigational safety challenges for USV. Figure 14(b) presents the improved A* algorithm, which incorporates a terrain safety expansion distance set to 9 units. Diagonal obstacle traversal is prohibited, while a smoothing algorithm simplifies path points. This creates a gap between the planned path and obstacle boundaries, providing navigational redundancy for the USV. Figure 14(c) depicts the A* algorithm employing Mamdani fuzzy inference. While maintaining the expanded terrain safety margin, it considers corner distances from obstacles to ensure navigational safety. However, constrained by corner limitations, the resulting path curves do not optimise the shortest distance; Figure 14(d) depicts the A*–fuzzy algorithm designed herein. It sets the terrain expansion safety margin to 9, employs T-S model fuzzy inference for obstacle spacing, and utilises RDF to simplify path points, achieving a balance between high navigational safety and path length.

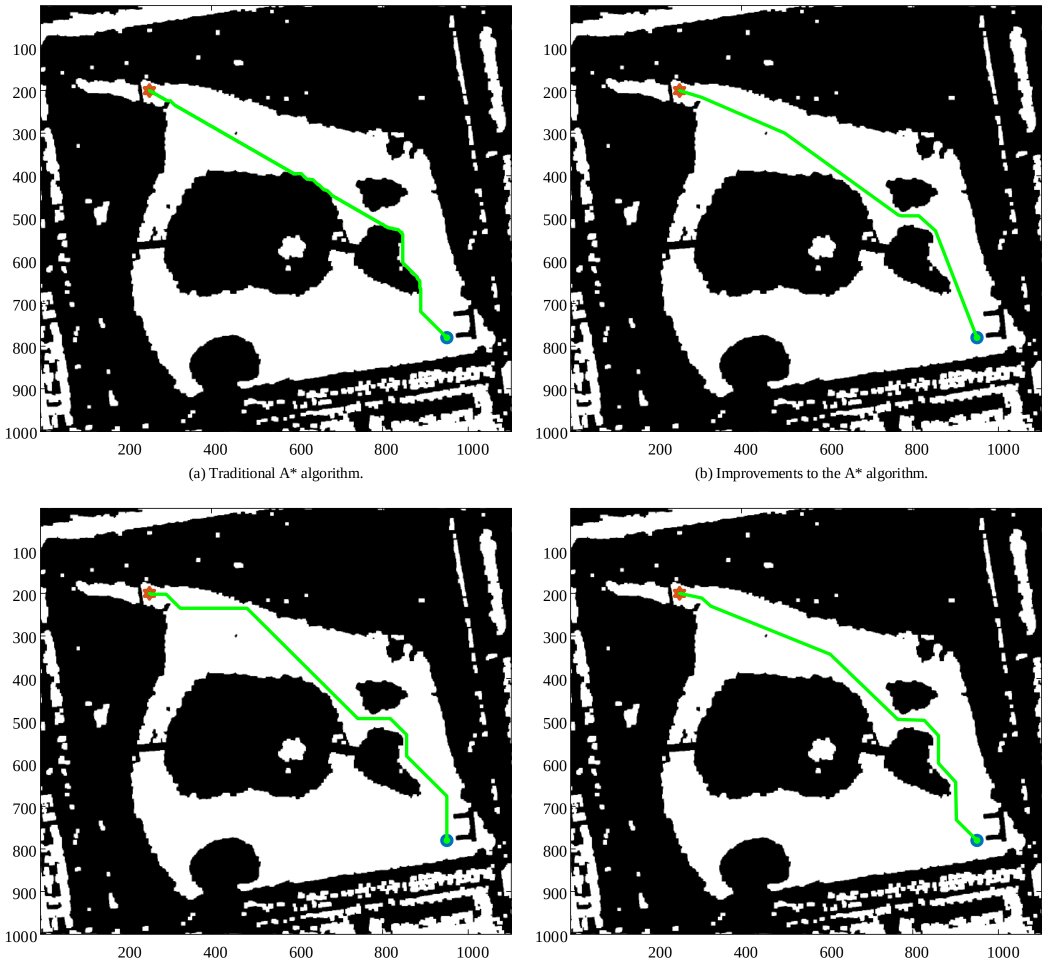


Figure 14: Search results and comparison of the path planning algorithms.

Table 3 summarizes the statistics of the planned-path results for each algorithm. The results show that the conventional A algorithm does not exhibit advantages across the evaluated criteria. The improved A* algorithm achieves fast computation, the shortest path length, and the fewest path points; however, it yields a smaller minimum clearance to obstacles (i.e., lower navigation safety) than the two fuzzy-inference-based approaches, because path-point simplification inevitably sacrifices part of the safety margin. The Mamdani fuzzy-inference A* algorithm ensures sufficient safety redundancy, but requires a longer planning time and produces relatively longer paths. In contrast, the proposed A*–fuzzy algorithm attains a runtime comparable to that of the improved A* algorithm while achieving the highest safety redundancy, which helps reduce the navigation risk of the USV, and it strikes a good balance between path length and the number of path points.

Table 3: Experimental comparison of path planning algorithms.

Statistical Path Data	Traditional A*	Improving A*	Fuzzy A* (Mamdani model)	This paper's A*–fuzzy algorithm
Time (s)	83.81	1.13	9.93	1.58
Path length (cells)	1018.43	962.00	1025.46	994.66
Number of waypoints	838	9	850	11
Minimum distance from obstacles (cells)	1.00	9.49	9.90	10.63

6. Conclusion

This paper presents an A*–fuzzy algorithm incorporating safety redundancy for collision avoidance, designed for practical USV navigation scenarios. The approach begins by selecting actual satellite imagery for binarization and raster processing, establishing safety thresholds for obstacle boundary expansion based on vessel dimensions and field test data. Subsequently, the T-S model is employed for fuzzy reasoning collision avoidance, leveraging fuzzy semantic experience to further distance vessels from obstacles and enhance navigational safety. Finally, the RDP algorithm simplifies and sparsifies the planned path curve while preserving original turning point tendencies and information content, yielding a path point set suitable for physical deployment. Simulation results demonstrate the designed algorithm's high safety redundancy, achieving balanced performance across time, path length, and path point count dimensions. It serves as a viable decision-support tool for practical unmanned vessel navigation planning.

References

[1] Rastkhiz E A, Schwartz H, Lambadaris I. A fuzzy set-based methodology for autonomous navigation[J]. Fuzzy Sets and Systems, 2025, 518: 109485.

[2] Xu Y, Liu W, Xu Y, et al. Enhanced A–Fuzzy DWA Hybrid Algorithm for AGV Path Planning in Confined Spaces[J]. World Electric Vehicle Journal, 2025, 16(9).*

[3] Ntakolia C, Lyridis D V. A Swarm Intelligence Graph-Based Pathfinding Algorithm Based on Fuzzy Logic (SIGPAF): A Case Study on Unmanned Surface Vehicle Multi-Objective Path Planning[J]. Journal of Marine Science and Engineering, 2021, 9(11): 1243.

[4] Airlangga G, Liu A. A Knowledge-Driven Approach to Dynamic Path Planning: Fuzzy A-Based Method for Scalable Multi-Agent Systems[C]//2023 IEEE Sixth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE). 2023: 71-74.*

[5] Sangeetha V, Ravichandran K S. A Modified Fuzzy A Based Inference System for Path Planning in an Unknown Environment[C]//2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI). 2018: 181-186.*

[6] Bantu A, Wiora J. Impact of Non-Gaussian Noise on Position Accuracy Provided by Global Navigation Satellite Systems (GNSS)[J]. IEEE Access, 2025, 13: 161750-161761.

[7] El-Ashmawy K L A. Vertical Accuracy of Google Earth Data[J]. Engineering, Technology & Applied Science Research, 2024, 14(3): 13830-13836.

[8] Potere D. Horizontal Positional Accuracy of Google Earth's High-Resolution Imagery Archive[J]. Sensors, 2008, 8(12): 7973-7981.

[9] Nwilo P C, Okolie C J, Onyegbula J C, et al. Positional accuracy assessment of historical Google Earth imagery in Lagos State, Nigeria[J]. Applied Geomatics, 2022, 14(3): 545-568.

[10] Takagi T, Sugeno M. Fuzzy identification of systems and its applications to modeling and control[J]. IEEE Transactions on Systems, Man, and Cybernetics, 1985, SMC-15(1): 116-132.

[11] Ramer U. An iterative procedure for the polygonal approximation of plane curves[J]. Computer Graphics and Image Processing, 1972, 1(3): 244-256.