An Integrated Framework for Safe and Efficient AUV Navigation: Synergizing Enhanced Path Planning, Curvature-Adaptive Tracking, and Information-Driven 3D Exploration

Mingming Xiao , Yuliang Wen , Jiaheng Li * , Naiyao Liang , Dan Xiang

Posted Date: 16 April 2026

doi: 10.20944/preprints202604.1168.v1

Keywords: autonomous navigation; finite-state machine; 3D autonomous exploration; speed control

 doi:10.20944/preprints202604.1168.v1

Article

An Integrated Framework for Safe and Efficient AUV Navigation: Synergizing Enhanced Path Planning, Curvature-Adaptive Tracking, and Information-Driven 3D Exploration

Mingming Xiao, Yuliang Wen, Jiaheng Li *, Naiyao Liang and Dan Xiang

School of Artificial Intelligence, Guangzhou Maritime University, Guangzhou 510725, China
* Correspondence: lijiaheng1009@163.com

Abstract

Efficient path planning and trajectory tracking are central to the safe and autonomous navigation of autonomous underwater vehicles (AUVs) in complex and unknown environments. To address the inherent challenges of safety, smoothness, and exploration efficiency in such settings, this paper presents an integrated framework that synergistically couples three enhanced core modules with complementary innovations. First, improved I-LazyTheta* and A-IRRT* algorithms are developed to incorporate safety margin collision detection and dynamic obstacle avoidance weight regulation, which enable efficient generation of collision-free paths that proactively maintain safe clearance in cluttered 3D spaces. Second, a trajectory tracking module based on a finite-state machine is designed, integrating B-spline optimization and a curvature-adaptive speed control mechanism to ensure high-precision following with guaranteed path smoothness and trackability. Third, a novel 3D autonomous exploration strategy tailored to underwater sonar constraints is constructed, combining frontier point clustering, multi-dimensional information gain evaluation, and traveling salesman problem (TSP) path optimization to achieve efficient unknown environment traversal while significantly reducing redundant detection and energy consumption. The proposed framework supports modular decoupling for independent reuse as well as integrated collaborative operation, offering flexible adaptability to diverse underwater robotic platforms. Simulations demonstrate that the integrated approach achieves superior performance in path safety and tracking accuracy, along with an exploration coverage of 79.08%, validating its effectiveness for robust AUV autonomy in complex underwater scenarios.

Keywords: autonomous navigation; finite-state machine; 3D autonomous exploration; speed control

1. Introduction

Autonomous underwater vehicles (AUVs) have attracted increasing attention in marine applications such as underwater detection, resource exploration, underwater search and rescue, and waterway inspection [1,2]. In unknown and unstructured 3D underwater environments, AUVs must possess strong autonomous decision-making and motion control capabilities. However, their 3D autonomous exploration faces multiple challenges. First, the scarcity of prior information about the underwater environment, where obstacle distributions are unknown and dynamically changing, requires path planning algorithms to have real-time obstacle avoidance and safety assurance capabilities [3,4]. Second, the strongly nonlinear dynamic characteristics of AUVs make it difficult to directly track purely geometric paths, necessitating the generation of executable trajectories that satisfy curvature continuity and dynamic constraints [5,6]. Third, viewpoint selection during the exploration process directly affects information gain efficiency, and unreasonable viewpoint traversal strategies can lead to redundant detection and energy waste [7,8]. Among these, path planning and trajectory tracking technologies are

 doi:10.20944/preprints202604.1168.v1

directly related to the operational efficiency and survival safety of the vehicle, serving as core elements for ensuring the completion of autonomous exploration tasks.

In recent years, scholars both in China and abroad have conducted an increasing amount of research on path planning algorithms for obstacle avoidance [9]. In terms of planning, sampling-based methods (e.g., Rapidly-exploring Random Tree (RRT), RRT*) explore the configuration space through random sampling, offering probabilistic completeness, yet suffer from slow convergence and tortuous, redundant paths [10,11]. Ying et al. improved the RRT algorithm through regional sampling, gravitational guidance, and bridge tests, reducing path length and runtime while lowering memory consumption and the number of nodes [12]. Li et al. incorporated rolling planning and node screening to improve the RRT algorithm, combining search decisions with trajectory prediction to achieve efficient target search and interception for AUVs in 3D environments [13]. Search-based methods (e.g., A*, Theta*) leverage heuristic functions to guide the search, making them suitable for low-dimensional spaces, but their computational cost increases significantly in 3D grid maps [14,15]. Aine et al. extended the search capability of the A* algorithm by integrating multiple heuristic functions, improving planning efficiency in complex environments [16]. Faria et al. extended the Lazy Theta* algorithm to sparse grid 3D spaces, achieving rapid path planning for unmanned aerial vehicles [17]. Chen et al. incorporated ocean current effects and constraints into A* path planning, generating feasible paths that conform to marine dynamic characteristics [18]. Although a large amount of literature has conducted research and optimization on path planning algorithms at present, most of the studies still remain in the scenarios of two-dimensional planes or those with known obstacle positions [19]. There is insufficient targeted analysis on the path planning of AUVs in complex 3D underwater environments, and generally, the robots are simplified to point masses for collision detection, ignoring the need for safety margins.

In trajectory optimization and tracking, B-spline curves are widely used for path smoothing due to their local support and high-order continuity [20,21]. Sun et al. applied uniform B-spline optimization to UAV trajectory generation, achieving smooth and dynamically feasible trajectories by combining the Limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm with a fast restart mechanism [22]. However, most B-spline-based methods are designed for aerial vehicles and rarely consider the inherent velocity-curvature coupling constraints in AUV hydrodynamics, which are critical for ensuring trajectory smoothness. In autonomous exploration, frontier detection is a classic strategy for exploring unknown environments [23]. Vidal et al. proposed a viewpoint generation method for underwater environments based on 2D frontiers, enabling AUVs to efficiently complete exploration and mapping tasks in planar environments [24]. However, this method only generates viewpoints in a two-dimensional plane and does not account for 3D sensor field-of-view coverage or volumetric information gain, making it difficult to meet the requirements of complete 3D underwater exploration tasks. Existing frontier-based methods generally lack sufficient consideration of practical underwater sensor constraints, viewpoint clustering, and information gain optimization, leaving considerable room for improvement in exploration efficiency.

To address the limitations of safety, smoothness, and exploration efficiency, this paper proposes a modular framework for autonomous exploration and trajectory tracking of AUVs in unknown 3D underwater environments, systematically solving the safety and efficiency challenges of autonomous navigation in complex scenarios without prior information. The core contributions of this paper are as follows:

- We propose I-LazyTheta* and A-IRRT* algorithms to integrate safety margin collision detection, obstacle avoidance weight regulation, and online replanning mechanisms to enable efficient online generation of safe and feasible paths for AUVs in complex 3D underwater environments.
- We propose a finite-state machine-based tracking module with B-spline optimization and curvature-adaptive deceleration to ensure trackable smooth trajectories, and a sonar-constrained exploration strategy combining frontier clustering, multi-dimensional gain evaluation, and TSP optimization to achieve efficient traversal of unknown underwater areas.

 doi:10.20944/preprints202604.1168.v1

- Simulations validate that the proposed framework and improved algorithms significantly outperform traditional RRT* in planning efficiency, path quality, tracking accuracy and exploration performance, and achieve 79.08% exploration coverage in unknown underwater environments.

The remainder of this paper is organized as follows. In Section 2, the fundamental theories and existing problems of traditional path planning algorithms are introduced. The design details of the proposed framework and algorithms are elaborated in Section 3. In Section 4, simulation experiments and result analysis are presented. Finally, the conclusions and future research directions are summarized in Section 5.

2. Traditional Path Planning Algorithms

Search-based path planning algorithms operate on grid maps and guide node exploration through cost functions. Representative algorithms include A*, Theta*, and LazyTheta*. The core cost function of the A* algorithm is expressed as follows:

$$F(n) = G(n) + H(n) \tag{1}$$

where $G(n)$ represents the actual movement cost from the start node to the current node n, typically computed using Euclidean distance, and $H(n)$ is the heuristic cost from the current node to the goal node [25,26].

The Theta* algorithm addresses the issue of redundant path corners in A* by introducing a line-of-sight (LOS) mechanism, allowing a node to directly connect to a non-adjacent parent node, as shown in Eq. (2). For the current expanded node n, if the LOS between its parent node p_n and n is clear, the cumulative cost $g(n)$ is updated directly using Euclidean distance; otherwise, the algorithm reverts to the traditional A* strategy, searching among the neighbor set $\mathcal{N}(n)$ for the predecessor node with the minimum cost:

$$g(n) = \begin{cases} g(p_n) + \|p_n - n\|_2, & \text{if LOS}(p_n, n) \\ \min\limits_{x \in \mathcal{N}(n)} \left(g(x) + \|x - n\|_2\right), & \text{otherwise} \end{cases} \tag{2}$$

where $\text{LOS}(a, b)$ is a Boolean function that only determines whether the line segment ab passes through any occupied grid cells.

LazyTheta* reduces the computational burden of Theta* by delaying LOS checks until node expansion. Its core logic remains consistent with Eq. (2), only adjusting the timing of LOS execution to reduce the number of checks. Leveraging the local optimization characteristics of line-of-sight detection, the algorithm balances search efficiency and path quality, making it widely applied to path planning problems in 3D environments [27]. However, its LOS detection lacks a result caching mechanism. In 3D grid scenarios, the collision status of the same node and the same line segment is repeatedly computed, causing the time complexity of the algorithm to grow exponentially with the number of grid cells, thereby failing to meet the real-time requirements of AUVs online planning.

Sampling-based path planning algorithms explore the configuration space through random sampling, with RRT* and Informed-RRT* being representative algorithms. The traditional RRT* algorithm progressively approaches the optimal path through random sampling and iterative path optimization, offering the advantage of probabilistic completeness. The core formulation of the RRT* algorithm is as follows:

$$g(\mathbf{x}_{\text{new}}) = \min_{\mathbf{x}_{\text{near}} \in \mathcal{X}_{\text{near}}} \left(g(\mathbf{x}_{\text{near}}) + \|\mathbf{x}_{\text{near}} - \mathbf{x}_{\text{new}}\|_2\right) \tag{3}$$

where $\mathcal{X}_{\text{near}}$ is the set of neighboring nodes of $\mathbf{x}_{\text{new}}$.

The algorithm achieves asymptotic optimality through parent re-selection and rewiring, with sampling performed over the entire configuration space [28]. However, its sampling process lacks spatial constraints. In complex 3D underwater environments with intricate obstacle distributions, the

 doi:10.20944/preprints202604.1168.v1

proportion of effective sampling points is low, and the number of iterations for path optimization is high, leading to a significant reduction in convergence rate. Consequently, a large number of samples are required to generate a high-quality path that meets the navigation requirements of AUVs.

The Informed-RRT* algorithm, as a typical improvement over RRT*, constructs an elliptical sampling domain with the start and goal as foci after obtaining an initial feasible path. The parameters for the ellipsoid semi-axes are given by:

$$r_1 = \frac{c_{\text{best}}}{2}, \quad r_2 = r_3 = \frac{\sqrt{c_{\text{best}}^2 - c_{\text{min}}^2}}{2} \tag{4}$$

where c_{best} is the length of the current optimal path, c_{min} is the Euclidean distance from the start to the goal, r_1 is the major axis, and r_2 and r_3 are the minor axes in the XY plane and along the Z-axis, respectively.

The Informed-RRT* algorithm constrains the sampling space from the full configuration space to the elliptical region, improving convergence efficiency by reducing ineffective sampling [29]. However, its sampling mode with fixed-parameter elliptical constraints does not account for the obstacle distribution characteristics of the underwater 3D environment or the motion constraints of the AUVs. In scenarios where obstacles block the direct path between the start and goal, it easily falls into local optima due to the over-constrained sampling space, making it difficult to find a globally optimal safe path.

Furthermore, the aforementioned traditional planning algorithms simplify the robot as a mass point in collision detection.They only verify whether path nodes lie inside occupied voxels, thus neglecting the robot's actual geometric dimensions. In unstructured environments such as underwater reef clusters and pipelines, this simplification can easily result in insufficient safety clearance between the planned path and obstacles, thereby posing collision risks during execution. In addition, these algorithms rely on static maps and lack periodic collision status verification or replanning triggers. Consequently, when the OctoMap is dynamically updated with sonar perception, the planned path may become invalid in the presence of newly detected obstacles, further compromising navigation safety.

3. The Proposed Secure and Efficient Integration Framework

In this section, we propose modular framework for AUVs 3D autonomous exploration, including the design and realization of improved path planning, trajectory optimization and tracking, and 3D autonomous exploration modules. It enable safe and efficient autonomous navigation for AUVs in unknown underwater environments.

3.1. Improved Path Planning Algorithm

As previously mentioned, traditional path planning algorithms for underwater 3D scenarios suffer from low search efficiency, poor path quality, and insufficient adaptability to dynamic environments. To address these issues, this paper proposes an improved path planning framework for AUVs, where the overall workflow is illustrated in Figure 1. The improved algorithms jointly introduce an obstacle avoidance weight mechanism to accommodate the directional distribution characteristics of underwater obstacles. They also employ multi-round greedy pruning combined with linear interpolation as a unified path post-processing strategy. Additionally, they support periodic collision detection and dynamic replanning to cope with environmental changes. Within this unified framework, we introduce targeted optimizations for Informed-RRT*. These include hybrid ellipsoidal sampling, adaptive iteration, and local planning modes. For LazyTheta*, we introduce a collision detection caching mechanism. Both algorithms jointly adopt the obstacle avoidance weight guidance and multi-round greedy pruning strategies, thereby significantly improving computational efficiency and environmental adaptability while ensuring planning quality.

 doi:10.20944/preprints202604.1168.v1

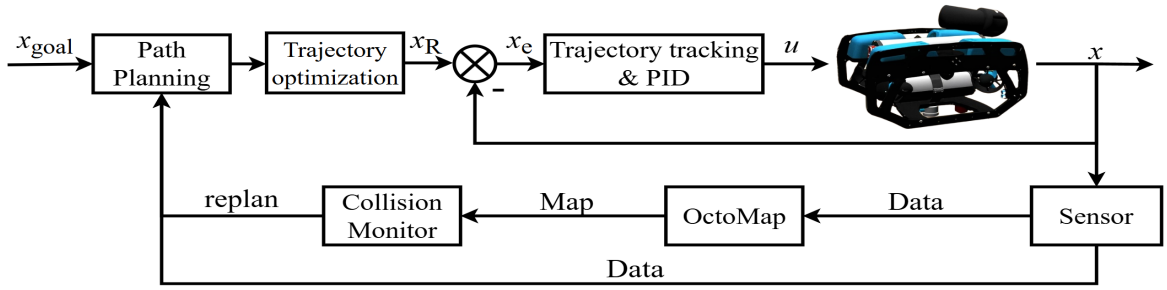


Figure 1. Flowchart of the improved path planning algorithm.

3.1.1. General Improvement Strategies

a) 3D Safety Margin Collision Detection Based on FCL: To address the issue of the planner neglecting the physical dimensions of the robot, this study adopts OctoMap as a probabilistic occupancy grid representation for the 3D environment [30], combined with the Flexible Collision Library (FCL) to perform efficient collision queries on OctoMap voxels. The BlueROV2 is modeled as a cuboid collision volume with dimensions (l, w, h), and a safety margin δ is applied. The actual collision envelope of the robot is given by:

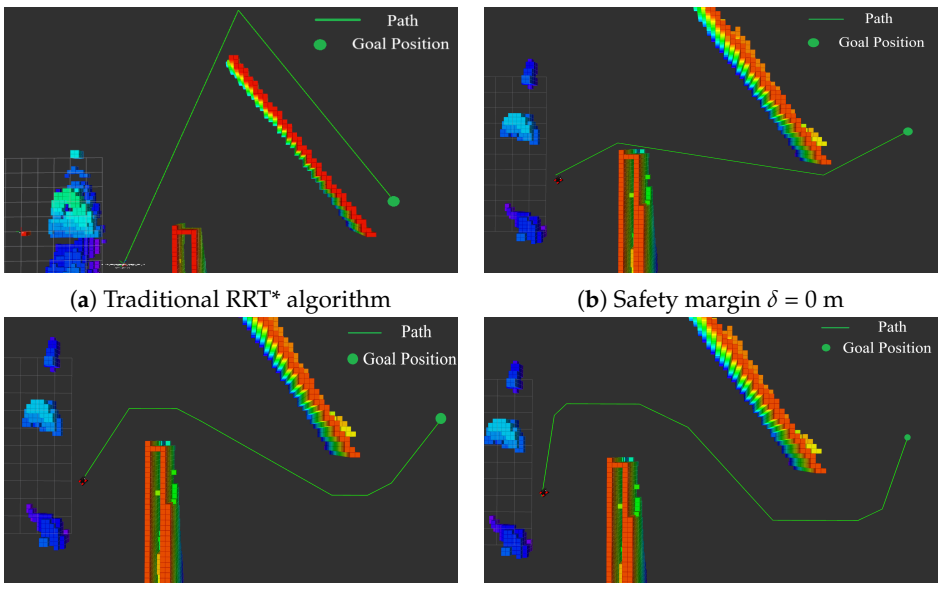
$$\mathbf{B} = (l + \delta) \times (w + \delta) \times (h + \delta) \tag{5}$$

Given the robot pose $(\mathbf{R}, \mathbf{t})$, the collision volume in the world coordinate frame is represented as:

$$\mathbf{B}_{\text{world}} = \mathbf{R} \cdot \mathbf{B} + \mathbf{t} \tag{6}$$

where $\mathbf{R}$ is the rotation matrix and $\mathbf{t}$ is the translation vector.

As shown in Figure 2, this improvement ensures that the planned path maintains a safety distance of no less than δ from obstacles at all times, guaranteeing that the path is collision-free under the dual assurance of the robot's actual geometric constraints and the safety margin. This fundamentally addresses the safety hazards caused by the mass point assumption.



(**a**) Traditional RRT* algorithm (**b**) Safety margin δ = 0 m

(**c**) Safety margin δ = 2.5 m (**d**) Safety margin δ = 5 m

Figure 2. Comparison of the improved algorithm and the traditional algorithm.

b) Online Monitoring and Replanning Based on OctoMap: Underwater environments are dynamic, and the OctoMap is continuously updated as sonar perception progresses. Traditional algorithms perform only a single offline planning and cannot perceive the collision status of the path after map

 doi:10.20944/preprints202604.1168.v1

updates. When a planned path becomes invalid due to the emergence of new obstacles, the system lacks an autonomous response mechanism, posing serious safety hazards. To address this, this paper designs an online trajectory monitoring and replanning module. This module periodically detects the collision status between the currently executed path and the latest OctoMap at a configurable frequency (default 2 Hz).

For a trajectory consisting of n waypoints $\{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$, the system uses the 3D Bresenham algorithm for discretized collision detection. For the line segment $\overline{\mathbf{p}_i\mathbf{p}_{i+1}}$, the algorithm samples uniformly along the segment. The positions of the sampling points are:

$$\mathbf{s}_k^{(i)} = \mathbf{p}_i + \frac{k}{K_i}(\mathbf{p}_{i+1} - \mathbf{p}_i), \quad k = 0, 1, \ldots, K_i \tag{7}$$

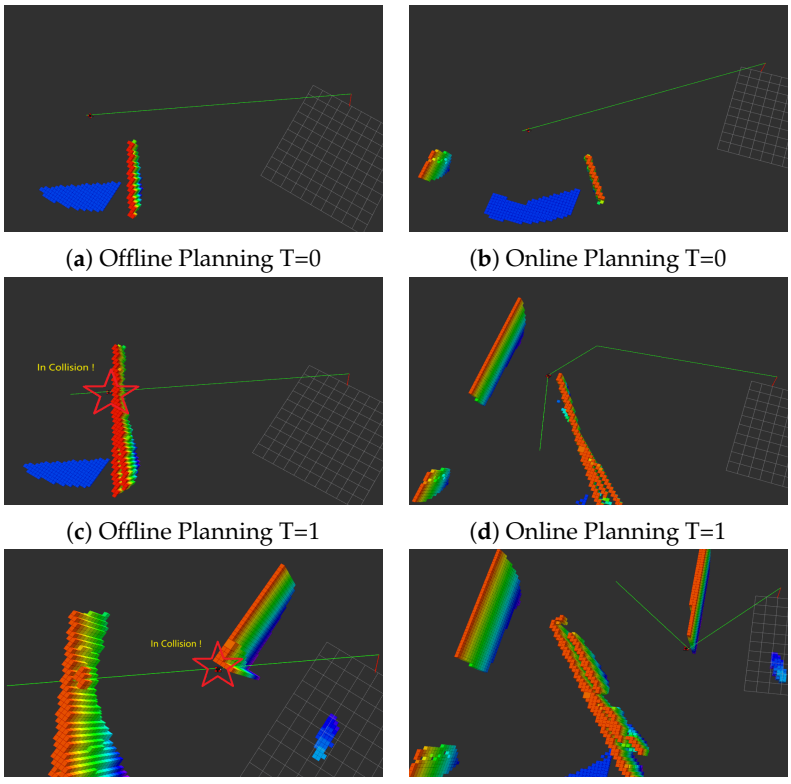
Here, the number of samples is $K_i = \left\lceil \frac{\|\mathbf{p}_{i+1} - \mathbf{p}_i\|}{\Delta s} \right\rceil$, and the sampling interval Δs is set according to the robot dimensions:

$$\Delta s = a \cdot \min(l + \delta, w + \delta, h + \delta) \tag{8}$$

In this study, $a = 0.7$. At each sampling point $\mathbf{s}_k^{(i)}$, the orientation is computed via spherical linear interpolation (slerp):

$$\mathbf{q}_k^{(i)} = \text{slerp}\left(\mathbf{q}_i, \mathbf{q}_{i+1}, \frac{k}{K_i}\right) \tag{9}$$

The robot collision volume is then constructed, centered at $\mathbf{s}_k^{(i)}$ with orientation $\mathbf{q}_k^{(i)}$ and dimensions $\mathbf{B}$. If the collision volume at any sampling point intersects with an obstacle, the path is considered to be in collision. The system immediately initiates replanning from the current robot position by invoking the corresponding planner to regenerate a safe and feasible path, thereby ensuring the safe navigation of the vehicle in complex dynamic environments. As shown in Figure 3, online path monitoring and replanning can effectively respond to dynamic updates in unknown environments.



(**a**) Offline Planning T=0 (**b**) Online Planning T=0

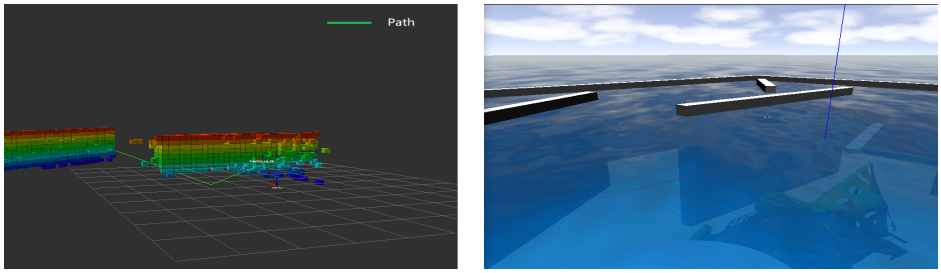
(**c**) Offline Planning T=1 (**d**) Online Planning T=1

(**e**) Offline Planning T=2 (**f**) Online Planning T=2

Figure 3. Examples of traditional offline planning and improved online planning in the same obstacle scenario. T = 0, 1, and 2 represent three different time steps.

 doi:10.20944/preprints202604.1168.v1

c) Obstacle Avoidance Weight Mechanism: The spatial distribution of underwater obstacles exhibits significant anisotropic characteristics: reef clusters typically extend horizontally, while structures such as pipelines and anchor chains may be vertically oriented. Moreover, limited by the forward-looking sonar field of view of the underwater vehicle, the visible range of obstacles ahead exhibits the characteristics shown in Figure 4. Traditional path planning algorithms, aiming to generate the shortest path, often tend to choose vertical detour paths around obstacles. However, if the underwater vehicle fails to detect newly identified obstacles in time, it is highly likely to cause collision risks.



(**a**) Constrained sonar FOV mapping (**b**) Actual obstacle environment

Figure 4. Traditional planning generates unsafe paths under sonar FOV constraints.

To address this issue, we introduce an obstacle avoidance weight parameter $w_{\text{obs}} \in [0,1]$. It uniformly regulates the tendency of the planning algorithm between horizontal and vertical obstacle avoidance. In practical application scenarios, when obstacles are predominantly vertically distributed, the parameter is set to a value close to 0, and the algorithm prioritizes horizontal obstacle avoidance strategies. Conversely, when obstacles are predominantly horizontally distributed, the parameter is set to a value close to 1, and the algorithm prioritizes vertical obstacle avoidance strategies. Through this parameter configuration, a safer path can be generated for the AUVs.

d) Path Post-Processing: Paths generated by traditional planning typically contain redundant waypoints with uneven density. Directly using such paths for trajectory optimization increases the subsequent computational burden. This paper adopts a multi-round greedy pruning strategy: first, starting from the initial point, it greedily skips as many intermediate nodes as possible; then, reverse optimization is performed from the goal point to further eliminate redundancy; finally, linear interpolation is applied to the pruned path to provide uniformly distributed waypoints for subsequent trajectory optimization. The number of interpolation points is determined by the path segment length and the maximum allowable path segment length $d_{\max}$:

$$N_{\text{interp}} = \left\lceil \frac{\|\mathbf{p}_{i+1} - \mathbf{p}_i\|}{d_{\max}} \right\rceil \tag{10}$$

The positions of the interpolated waypoints are given by:

$$\mathbf{p}(t) = \mathbf{p}_i + t(\mathbf{p}_{i+1} - \mathbf{p}_i), \quad t \in \left\{ \frac{k}{N_{\text{interp}}} \right\}_{k=1}^{N_{\text{interp}}} \tag{11}$$

3.1.2. A Planner with Targeted Improvements

a) The Informed-RRT* Algorithm with Adaptive Sampling (A-IRRT*): The traditional Informed-RRT* algorithm uses fixed elliptical parameters that cannot dynamically adjust the sampling range based on the obstacle distribution in the underwater environment, making it difficult to achieve a balance between sampling efficiency and global search capability. On the other hand, it does not incorporate dimension sampling preferences tailored to the obstacle avoidance requirements of AUVs 3D navigation, failing to adapt to the anisotropic characteristics of horizontally or vertically distributed underwater obstacles.

To address these issues, we propose targeted improvements and theoretical optimizations to the Informed-RRT* algorithm. These combine the topological characteristics of the underwater 3D

 doi:10.20944/preprints202604.1168.v1

environment with the dynamic navigation constraints of the AUVs. The core contributions are twofold: First, we integrate general improvement strategies such as safety margin collision detection and obstacle avoidance weight mechanism into the algorithm framework. This ensures that the sampling and path generation process fully satisfies the geometric constraints and safety obstacle avoidance requirements of the AUVs. Second, we propose a stage-adaptive hybrid sampling strategy and a distance-adaptive iteration control mechanism. In the early stage of path search, it adopts a hybrid mode of relaxed ellipsoidal sampling and free space sampling to expand the effective exploration range. After obtaining an initial path, it switches to adaptive-parameter ellipsoidal sampling, while dynamically adjusting the maximum number of iterations based on the Euclidean distance from the start to the goal. This achieves a dynamic balance between sampling efficiency and path search completeness. The improved adaptive Informed-RRT* algorithm not only solves the problem of traditional Informed-RRT* easily falling into local optima but also further enhances the convergence rate in complex underwater 3D environments. It enables more efficient generation of initial feasible paths that satisfy safety constraints for AUVs. The specific implementation is as follows.

During the initial phase of path search, when no feasible path has been found, this design adopts a hybrid sampling strategy, performing relaxed ellipsoidal sampling with a probability of 60% and free space sampling with a probability of 40%. The initial major axis of the relaxed ellipsoid is set to $c = 2.5 \times \|\mathbf{x}_{\text{goal}} - \mathbf{x}_{\text{start}}\|$, i.e., 2.5 times the Euclidean distance from the start to the goal, to expand the early exploration range. Free space sampling ensures that the algorithm can discover feasible regions behind obstacles. Once an initial path is found, the algorithm switches to the standard Informed sampling mode.

This study incorporates the obstacle avoidance weight parameter w_{obs} into the planning algorithm to adjust the algorithm's preference for obstacle avoidance in the horizontal and vertical directions. This parameter controls the shape of the ellipsoid along the Z-axis (vertical direction), thereby enabling dimensionally adaptive sampling. The sampling points are transformed from the ellipsoid coordinate frame to the world coordinate frame via the rotation matrix $\mathbf{R}$:

$$\mathbf{p}_{\text{world}} = \mathbf{R} \begin{bmatrix} r_1 \cdot x_{\text{ball}} \\ r_2 \cdot y_{\text{ball}} \\ r_3 \cdot z_{\text{ball}} \cdot w_{\text{obs}} \end{bmatrix} + \frac{\mathbf{p}_{\text{start}} + \mathbf{p}_{\text{goal}}}{2} \tag{12}$$

where $x_{\text{ball}}, y_{\text{ball}}, z_{\text{ball}} \in [-1, 1]$ are random sampling points within a unit sphere.

When $w_{\text{obs}} = 0$, the vertical (Z-axis) sampling is compressed to zero, and the ellipsoid degenerates into a horizontal two-dimensional ellipse, prioritizing the search for planar paths to avoid vertical perception blind spots. When $w_{\text{obs}} = 1$, the ellipsoid maintains its full 3D shape, allowing full utilization of vertical space for obstacle avoidance.

To balance computational efficiency and path quality, the maximum number of iterations is dynamically adjusted based on the distance between the start and goal points:

$$N = N_{\min} + (N_{\max} - N_{\min}) \cdot \min\left(1, \frac{d_{\text{start-goal}}}{d_{\max}}\right) \tag{13}$$

where $N_{\min}$ is the minimum number of iterations, $N_{\max}$ is the maximum number of iterations, and $d_{\max}$ is a distance threshold.

For short-distance planning tasks, the algorithm uses fewer iterations to improve real-time performance. For long-distance tasks, the number of iterations is increased to ensure path quality. Simultaneously, to enhance the connectivity of explored regions, the planner incorporates a local exploration mechanism into the sampling strategy, performing local sampling near existing nodes with a probability of 20%. This strategy facilitates denser connections around obstacle boundaries, thereby improving path quality.

 doi:10.20944/preprints202604.1168.v1

Algorithm 1: A-IRRT*

Input: $\mathbf{p}_s$, $\mathbf{p}_g$, $\mathcal{M}$, w, $\mathcal{B}$
Output: Path π

if FCLCHECK($\mathbf{p}_s \to \mathbf{p}_g$) = *free* **then**
  return LINEARINTERPOLATE($\mathbf{p}_s$, $\mathbf{p}_g$);
$N_{iter} \leftarrow N_{min} + (N_{max} - N_{min}) \cdot \min(d(\mathbf{p}_s, \mathbf{p}_g)/d_{max}, 1)$;
$\mathcal{T} \leftarrow \{\mathbf{p}_s\}$, $c_{best} \leftarrow \infty$, *found* $\leftarrow$ **false**;
for $i \leftarrow 1$ **to** N_{iter} **do**
  if *found* **then**
    $\mathbf{x}_{rand} \leftarrow$ SAMPLEELLIPSOID(c_{best}, $r_z = r_2 \cdot w$);
  else
    $\mathbf{x}_{rand} \leftarrow$ MIXEDSAMPLE(c_{min}, w, $\mathcal{B}$);
  $\mathbf{x}_{new} \leftarrow$ STEER(NEAREST($\mathcal{T}, \mathbf{x}_{rand}$), $\mathbf{x}_{rand}$);
  if FCLCHECK($\mathbf{x}_{new}$) = *free* **then**
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathbf{x}_{new}\}$ via CHOOSEPARENT and REWIRE;
    if $d(\mathbf{x}_{new}, \mathbf{p}_g) < \epsilon$ **then**
      $c_{best} \leftarrow \min(c_{best}, \text{cost}(\mathbf{x}_{new}))$, *found* $\leftarrow$ **true**;
if *found* **then**
  $\pi \leftarrow$ EXTRACTPATH($\mathcal{T}$, $\mathbf{p}_g$);
  return PRUNE(π) $\to$ INTERPOLATE(π, Δl_{max});
return false;

b) Improved Lazy Theta* Algorithm (I-LazyTheta*): When the traditional LazyTheta* algorithm is applied to 3D underwater environments based on OctoMap, it requires performing FCL bounding box collision detection on a large number of grid nodes. Due to the high computational complexity of the FCL collision query mechanism based on convex hull intersection, combined with the fact that nodes in 3D grid scenes are frequently revisited during multi-path search processes, a significant number of redundant collision detection operations occur, resulting in ineffective consumption of computational resources [31]. This problem is particularly prominent in dense obstacle scenarios and becomes the main bottleneck restricting the planning speed of the algorithm. Furthermore, the floating-point computation mode employed in traditional line-of-sight detection further increases computational overhead and reduces the overall efficiency of path search.

To address the above issues, we propose targeted improvements and optimizations to the Lazy Theta* algorithm. These combine the grid map characteristics of the 3D underwater environment with the search properties of Lazy Theta*. The core contributions are twofold. First, based on integrating general improvement strategies such as safety margin collision detection and obstacle avoidance weight mechanism, we propose a hash table-based collision detection result caching mechanism. Using the grid coordinate triplet (g_x, g_y, g_z) as the unique key, the completed FCL collision query results are stored in a hash table. For repeatedly visited nodes, the cached results are directly retrieved, thereby completely eliminating the computational overhead of redundant collision detection. The cache is cleared before each planning invocation to ensure consistency between the cached results and the dynamically updated state of the current OctoMap. Second, we introduce the 3D Bresenham algorithm into the line-of-sight detection process. It utilizes integer operations to replace traditional floating-point computations, enabling rapid line-of-sight occlusion judgment in discrete grids. This significantly reduces the computational complexity of line-of-sight detection [32]. In dense obstacle 3D underwater scenarios, the improved Lazy Theta* algorithm substantially reduces the total time consumption of collision detection. It also enhances the computational efficiency of line-of-sight detection, thereby achieving a significant increase in planning speed. This better adapts to the real-time requirements of AUVs 3D path planning. The specific implementation is as follows.

 doi:10.20944/preprints202604.1168.v1

Algorithm 2: I-LazyTheta*

Input: $\mathbf{p}_s$, $\mathbf{p}_g$, $\mathcal{M}$, w, $\mathcal{B}$
Output: Path π

$\mathcal{B}_{dyn} \leftarrow \text{BoundingBox}(\mathbf{p}_s, \mathbf{p}_g) \oplus \Delta m$, $\mathcal{C} \leftarrow \emptyset$;
$\mathcal{O} \leftarrow \{\mathbf{p}_s\}$, $g(\mathbf{p}_s) \leftarrow 0$;
while $\mathcal{O} \neq \emptyset$ **do**
  $s \leftarrow \arg\min_{n \in \mathcal{O}} f(n)$, $\mathcal{O} \leftarrow \mathcal{O} \setminus \{s\}$;
  `// Lazy line-of-sight check`
  if $\neg$LineOfSight($s.parent$, s) **then**
    $s.parent \leftarrow \arg\min_{n \in \mathcal{N}(s)} [g(n) + c(n, s)]$;
  if $s = \mathbf{p}_g$ **then**
    $\pi \leftarrow$ ExtractPath(s);
    return Prune(π) $\rightarrow$ InterpSLERP(π, Δl_{max});
  foreach $n \in \mathcal{N}_{26}(s)$ **do**
    $occ \leftarrow \mathcal{C}[n]$ **if** $n \in \mathcal{C}$ **else** FCLCheck(n);
    if $occ = free$ **then**
      $h(n) \leftarrow d_{xy}(n, \mathbf{p}_g) \cdot (10w+1) + d_z(n, \mathbf{p}_g) \cdot (10(1-w)+1)$;
      UpdateVertex(s, n);
return false;

The obstacle avoidance weight parameter w_{obs} is integrated into the planning algorithm, and a heuristic function that decouples the horizontal and vertical directions is designed. Let the current node be $\mathbf{n} = (x_n, y_n, z_n)$ and the goal node be (x_g, y_g, z_g). The weighted heuristic function is given by:

$$h(\mathbf{n}) = d_{\text{horizontal}} \cdot p_h + d_{\text{vertical}} \cdot p_v \tag{14}$$

where the horizontal and vertical distances and weights are defined as:

$$\begin{cases} d_{\text{horizontal}} = \sqrt{(x_g - x_n)^2 + (y_g - y_n)^2} \\ d_{\text{vertical}} = |z_g - z_n| \\ p_h = 1 + 10 \cdot w_{\text{obs}} \\ p_v = 1 + 10 \cdot (1 - w_{\text{obs}}) \end{cases} \tag{15}$$

Here, $w_{\text{obs}} \in [0, 1]$ is the obstacle avoidance preference parameter. When $w_{\text{obs}} = 0$, the cost of vertical movement is amplified by a factor of 11, and the algorithm tends to circumvent obstacles in the horizontal direction. When $w_{\text{obs}} = 1$, the cost of horizontal movement is amplified, and the algorithm tends to utilize vertical space for obstacle avoidance.

To maintain consistency between the heuristic function and the cost function, the movement cost from node $\mathbf{n}_1 = (x_1, y_1, z_1)$ to node $\mathbf{n}_2 = (x_2, y_2, z_2)$ is defined as:

$$c(\mathbf{n}_1, \mathbf{n}_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \cdot p_h + |z_2 - z_1| \cdot p_v \tag{16}$$

3.2. Trajectory Optimization and Tracking Algorithms

For the discrete path generated by the planner, the system first performs reference trajectory optimization, and then achieves precise control through a trajectory tracking algorithm. The velocity control commands are sent to the ArduSub PID controller via MAVROS to drive the underlying motors. The trajectory tracker is designed using a finite state machine, comprising four states: idle, tracking, stopping, and reached, enabling full-process management of trajectory tracking. The state machine design diagram is shown in Figure 5.

 doi:10.20944/preprints202604.1168.v1

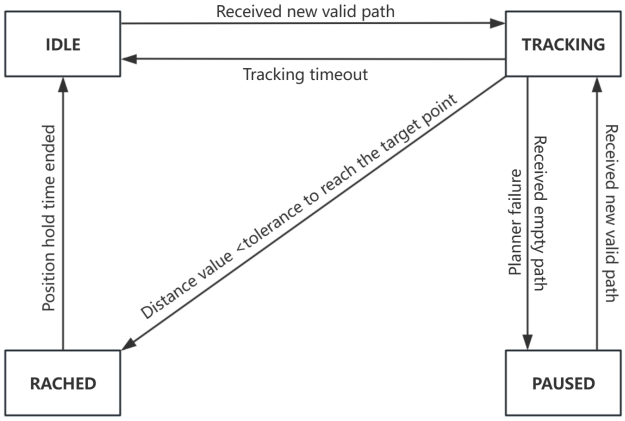


Figure 5. State machine design diagram for trajectory tracker.

3.2.1. B-Spline-Based Trajectory Optimization and Hermite Interpolation Tracking

A third-order B-spline curve is employed to fit the discrete path consisting of n waypoints, generating a smooth reference trajectory. The expression of the third-order B-spline curve is given by:

$$\mathbf{C}(u) = \sum_{i=0}^{n} \mathbf{Q}_i N_{i,3}(u), \quad u \in [0,1] \tag{17}$$

where $\mathbf{Q}_i$ are the path control points, and $N_{i,3}(u)$ are the third-order B-spline basis functions.

To ensure uniform distribution of trajectory points, the total arc length L of the curve is first estimated through 200 initial sampling points. Based on a desired waypoint spacing of 0.5 m, the target number of sampling points is calculated and constrained to be between 20 and 200. This adaptive strategy avoids both oversampling and undersampling issues. Simultaneously, to ensure the safe and stable operation of the moving vehicle on paths with varying curvature, the curvature at each point is estimated by calculating the change in direction between adjacent points, enabling adaptive speed limiting:

$$v_{\max,i} = \begin{cases} v_{\max}, & \kappa_i \le \kappa_{\text{th}} \\ \max\left(\dfrac{v_{\max}}{2}, \dfrac{v_{\max}}{1+\alpha\kappa_i}\right), & \kappa_i > \kappa_{\text{th}} \end{cases} \tag{18}$$

where $v_{\max}$ is the maximum allowable speed, $\kappa_{\text{th}} = 0.1$ is the curvature threshold, and $\alpha = 5$ is the decay factor.

In high-curvature regions, the speed is appropriately reduced to avoid collisions or loss of control caused by excessively sharp turns. On this basis, a bidirectional propagation optimization method is employed to achieve time-optimal velocity planning. Given the distance Δs_i between adjacent points, the forward and backward velocities at the current moment are computed as:

$$\begin{aligned} v_i^+ &= \min\left(\sqrt{v_{i-1}^2 + 2a_{\max}\Delta s_i},\ v_{\max,i}\right) \\ v_i &= \min\left(v_i^+,\ \sqrt{v_{i+1}^2 + 2a_{\max}^{\text{dec}}\Delta s_{i+1}}\right) \end{aligned} \tag{19}$$

where $a_{\max}$ and $a_{\max}^{\text{dec}}$ are the maximum acceleration and maximum deceleration, respectively.

To ensure C^2 continuity of the trajectory, i.e., continuous position, velocity, and acceleration, the tracker employs cubic Hermite interpolation. Given a time interval $[t_i, t_{i+1}]$, a normalized parameter $\tau = (t - t_i)/\Delta t$ is introduced. The basis functions of cubic Hermite interpolation are:

$$\begin{aligned} h_0(\tau) &= 2\tau^3 - 3\tau^2 + 1, & h_1(\tau) &= \tau^3 - 2\tau^2 + \tau, \\ h_2(\tau) &= -2\tau^3 + 3\tau^2, & h_3(\tau) &= \tau^3 - \tau^2. \end{aligned} \tag{20}$$

 doi:10.20944/preprints202604.1168.v1

Given the endpoint states (position $\mathbf{p}_i$ and velocity $\dot{\mathbf{p}}_i$), the interpolated position, velocity, and acceleration are:

$$\begin{aligned}\mathbf{p}(t) &= h_0(\tau)\mathbf{p}_i + h_2(\tau)\Delta t\dot{\mathbf{p}}_i + h_1(\tau)\mathbf{p}_{i+1} + h_3(\tau)\Delta t\dot{\mathbf{p}}_{i+1}\\ \dot{\mathbf{p}}(t) &= h_0'(\tau)\mathbf{p}_i + h_2'(\tau)\Delta t\dot{\mathbf{p}}_i + h_1'(\tau)\mathbf{p}_{i+1} + h_3'(\tau)\Delta t\dot{\mathbf{p}}_{i+1}\\ \ddot{\mathbf{p}}(t) &= h_0''(\tau)\mathbf{p}_i + h_2''(\tau)\Delta t\dot{\mathbf{p}}_i + h_1''(\tau)\mathbf{p}_{i+1} + h_3''(\tau)\Delta t\dot{\mathbf{p}}_{i+1}\end{aligned} \tag{21}$$

By incorporating velocity information, cubic Hermite interpolation ensures continuity of acceleration, enabling the underwater vehicle to operate more smoothly during trajectory tracking, reducing shocks and vibrations caused by trajectory discontinuities, and improving system stability and reliability.

3.2.2. Polynomial-Based Trajectory Optimization and Pure Pursuit Tracking

For a trajectory consisting of M segments, each segment is represented by a 7th-order polynomial:

$$\mathbf{p}_j(t) = \sum_{k=0}^{7} \mathbf{c}_{j,k} t^k \tag{22}$$

The optimization objective is to minimize snap:

$$\min J = \sum_{j=1}^{M} \int_0^{T_j} \|\mathbf{p}_j^{(4)}(t)\|^2 dt \tag{23}$$

Constraints include boundary constraints, waypoint constraints, and 4th-order continuity constraints. By employing an efficient QP solver (such as OSQP), the optimal coefficient solution can be obtained quickly, achieving a globally optimal smooth trajectory that satisfies all constraints.

The traditional Pure Pursuit algorithm uses a fixed look-ahead distance L_d, which tends to cause cutting corners on sharp turns, leading to trajectory tracking errors. This paper constructs a piecewise function to dynamically adjust the look-ahead distance based on the dot product $\rho = \mathbf{d}_1 \cdot \mathbf{d}_2$ of the direction vectors $\mathbf{d}_1$ and $\mathbf{d}_2$ of adjacent path points:

$$L_d = L_0 \times f(\rho) \tag{24}$$

where L_0 is the nominal look-ahead distance representing the base look-ahead distance on straight or low-curvature paths. In this paper's experiments, L_0 is set to 0.5 m. The scaling factor $f(\rho)$ is defined as:

$$f(\rho) = \begin{cases} 0.10, & \rho < 0 \\ 0.25 + 0.5\rho, & 0 \le \rho < 0.5 \\ \rho, & 0.5 \le \rho \le 1 \end{cases} \tag{25}$$

Based on the same curvature feature extraction mechanism, adaptive speed regulation on turning path segments is achieved. The base velocity v_{base} is dynamically generated by the acceleration/deceleration profile, ensuring a smooth transition between the starting acceleration segment and the terminal deceleration segment. The speed regulation function is designed as:

$$v_d = v_{\text{base}} \times \begin{cases} 0.30, & \rho < 0 \\ 0.30 + 0.4\rho, & 0 \le \rho < 0.5 \\ \rho, & 0.5 \le \rho \le 1 \end{cases} \tag{26}$$

This strategy reduces the speed to 30% of the base velocity on sharp turns, effectively mitigating trajectory deviations caused by centrifugal forces. On straight segments, it maintains over 50% of the speed, balancing tracking efficiency and motion smoothness.

 doi:10.20944/preprints202604.1168.v1

3.3. 3D Autonomous Exploration Mapping

The autonomous exploration system is primarily divided into three modules: information gain evaluation, frontier point detection and clustering, and TSP-based path optimization. The specific workflow is shown in Figure 6.

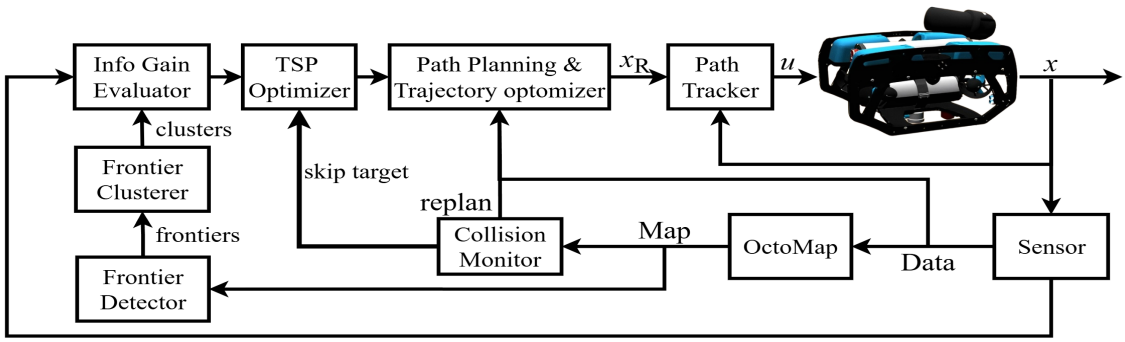


Figure 6. Control flowchart for autonomous exploration mode.

3.3.1. Information Gain Evaluation

In underwater autonomous exploration tasks, the robot needs to select the optimal exploration target from the set of detected frontier clusters $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$. Traditional methods typically consider only a single factor such as information gain or distance cost, making it difficult to balance exploration efficiency and path cost. This paper comprehensively considers three dimensions: base information gain, sonar visibility gain, and path cost. The comprehensive scoring function is defined as:

$$\begin{aligned} \text{Score}(C_i) = w_{\text{gain}} \cdot \text{IG}(C_i) + w_{\text{vis}} \cdot \text{VG}(C_i) \\ - w_{\text{dist}} \cdot \text{PC}(C_i) \end{aligned} \tag{27}$$

Here, $\text{IG}(C_i)$ represents the base information gain of frontier cluster C_i, quantifying the fundamental information value contained in the cluster. $\text{VG}(C_i)$ denotes the visibility gain, reflecting the detection capability of the sonar device for the cluster. $\text{PC}(C_i)$ is the path cost, i.e., the cost required for the robot to reach the cluster. w_{gain}, w_{vis}, and w_{dist} are the weight coefficients corresponding to the above factors, respectively, satisfying $w_{\text{gain}}, w_{\text{vis}}, w_{\text{dist}} \in [0, 1]$.

For a given cluster C_i, let its centroid coordinates be $\mathbf{c}_i$. We count the number of unknown voxels within a spherical search region $\mathcal{B}(\mathbf{c}_i, r_{\text{search}})$ centered at $\mathbf{c}_i$ with radius r_{search}:

$$\text{IG}(C_i) = \frac{1}{V_{\text{search}}} \sum_{\mathbf{v} \in \mathcal{B}(\mathbf{c}_i, r_{\text{search}})} \mathbf{I}(\mathbf{v} \in \mathcal{U}) \tag{28}$$

where $\mathcal{U}$ represents the set of unknown voxels, and $\mathbf{I}(\cdot)$ is the indicator function. $V_{\text{search}} = \frac{4}{3}\pi r_{\text{search}}^3$ is the volume of the spherical search region.

The introduction of the normalization factor V_{search} aims to confine the range of the base information gain to the interval $[0, 1]$. This treatment allows the base information gain to be reasonably weighted and fused with other indicators that have different dimensions and numerical ranges (such as visibility gain and path cost), thereby constructing a comprehensive evaluation system. In the simulation implementation, we set $r_{\text{search}} = 2.0\,\text{m}$ to balance computational efficiency and evaluation accuracy.

In traditional information gain evaluation methods, sensors are typically assumed to possess omnidirectional observation capabilities. However, in underwater autonomous exploration scenarios, sonar sensors have significant FOV limitations. If these physical constraints are ignored during information gain evaluation, the robot is highly likely to select target points with poor actual observability, leading to a substantial decrease in exploration efficiency [33].

 doi:10.20944/preprints202604.1168.v1

To address the above issue, a visibility gain model based on the sonar FOV is designed. This model employs a ray casting algorithm by emitting multiple rays within the sonar FOV and counting the number of unknown voxels observable by each ray, thereby enabling a quantitative evaluation of the visibility gain.

Let the current position of the robot be $\mathbf{p}_{\text{robot}}$. The Euclidean distance is adopted as the estimate of the path cost:

$$\text{PC}(C_i) = \frac{\|\mathbf{c}_i - \mathbf{p}_{\text{robot}}\|}{d_{\max}} \tag{29}$$

where the normalization factor $d_{\max}$ is set to five times the maximum effective detection range $r_{\max}$ of the sonar. This design ensures that the path cost is within the range $[0, 1]$, maintaining consistent dimensionality with the information gain metric.

To avoid redundant exploration of already explored areas and improve exploration efficiency, the system maintains a visited target point list $\mathcal{V}$. A first-in-first-out (FIFO) strategy is adopted to manage the visited target points. When the number of visited target points in the list reaches the maximum limit, the newly visited target point replaces the earliest one in the list. During the target selection process, for each candidate cluster C_i, the distance between its centroid $\mathbf{c}_i$ and every point $\mathbf{v}$ in the visited target point list $\mathcal{V}$ is calculated. If there exists any visited point $\mathbf{v}$ such that the distance between $\mathbf{c}_i$ and $\mathbf{v}$ is less than a preset exclusion radius $r_{\text{exclusion}}$, the candidate cluster is considered to be near an already explored area, and the system skips this cluster, no longer evaluating it as a candidate target. Through this strategy, the robot's repetitive exploration behavior is effectively avoided, ensuring the efficiency and comprehensiveness of the exploration process.

3.3.2. Frontier Detection and Clustering

Frontier points are the core input units of an autonomous exploration system, defined as voxels located at the boundary between known free space and unknown regions. In the 3D probabilistic map constructed based on OctoMap, a voxel must simultaneously satisfy the following conditions to be considered a frontier point: it must be marked as free and traversable; the number of unknown voxels within its 26-neighborhood must not be less than a preset minimum unknown neighbor threshold; and the shortest distance to all occupied voxels must not be less than a set safety distance threshold [34].

To reduce the computational burden, a voxel grid downsampling method is employed to decrease the number of frontier points. Within each downsampled voxel, only the frontier point with the largest number of unknown neighbors is retained as a representative point. The frontier points obtained through detection typically exhibit a clustered spatial distribution. To aggregate these adjacent frontier points into candidate regions for exploration targets for subsequent unified evaluation, the system employs the DBSCAN algorithm for clustering analysis [35], with a clustering distance threshold set to 2.0 m and a minimum points threshold set to 1, adapting to the sparse distribution of frontier points. After clustering, the system filters out small clusters with few points to eliminate noise and isolated point interference, and applies K-means segmentation to oversized clusters to prevent the increased complexity in subsequent evaluation and planning caused by a single excessively large cluster.

For each clustered cluster C_i, its centroid $\mathbf{c}_i$ serves as the representative position of the cluster for subsequent information gain evaluation and path planning. The centroid is calculated as the arithmetic mean of the coordinates of all points within the cluster:

$$\mathbf{c}_i = \frac{1}{|C_i|} \sum_{\mathbf{p} \in C_i} \mathbf{p} \tag{30}$$

where $\mathbf{p} = (x, y, z)^{\mathrm{T}}$ is a frontier point in C_i.

3.4. TSP Path Optimization

In multi-objective exploration scenarios, the robot needs to visit multiple frontier clusters sequentially. To minimize the total travel distance, we model the target ordering problem as a Traveling

 doi:10.20944/preprints202604.1168.v1

Salesman Problem (TSP). Let the current position of the robot be the starting point $\mathbf{p}_0$, and there exist n target points $\{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$. The core objective of TSP is to find the shortest path π that visits all target points, which is mathematically expressed as follows:

$$\min_{\pi \in \Pi} L(\pi) = \sum_{i=0}^{n-1} d(\mathbf{p}_{\pi(i)}, \mathbf{p}_{\pi(i+1)}) \tag{31}$$

where Π represents the set of all possible permutations of the target points. TSP is an NP-hard problem; if an exact solution method is adopted, its time complexity is as high as $O(n!)$. To achieve rapid exploration, we employ a greedy algorithm to generate an initial solution and use the 2-opt local optimization algorithm to improve the path quality.

The greedy algorithm is a heuristic algorithm based on local optimal selection. Its fundamental idea is to start from the starting point and, at each step, select the unvisited point closest to the current position as the next visiting point until all target points have been visited. The 2-opt algorithm is a classic local search algorithm for TSP, whose core idea is to improve the quality of the current solution by reversing subsequences within the path [36,37].

Given the current path π, the 2-opt algorithm attempts to find two edges $(\mathbf{p}_{\pi(i)}, \mathbf{p}_{\pi(i+1)})$ and $(\mathbf{p}_{\pi(j)}, \mathbf{p}_{\pi(j+1)})$ in the path, and calculates the improvement obtained by swapping these two edges:

$$\begin{aligned} \Delta = {} & d(\mathbf{p}_{\pi(i)}, \mathbf{p}_{\pi(i+1)}) + d(\mathbf{p}_{\pi(j)}, \mathbf{p}_{\pi(j+1)}) \\ & - d(\mathbf{p}_{\pi(i)}, \mathbf{p}_{\pi(j)}) - d(\mathbf{p}_{\pi(i+1)}, \mathbf{p}_{\pi(j+1)}). \end{aligned} \tag{32}$$

If $\Delta > 0$, it indicates that swapping these two edges will shorten the path length, and the swap operation is performed by reversing the subsequence from $\pi(i+1)$ to $\pi(j)$ in the path, yielding a new path π':

$$\begin{aligned} \pi' = {} & [\pi(0), \ldots, \pi(i), \pi(j), \pi(j-1), \ldots, \\ & \pi(i+1), \pi(j+1), \ldots, \pi(n)]. \end{aligned} \tag{33}$$

The 2-opt algorithm continuously iterates through the above operations to improve the path quality. To prevent the algorithm from falling into an infinite loop or excessive computation time, the maximum number of iterations $K_{\max} = 1000$ and the maximum running time $T_{\max} = 1.0\,\mathrm{s}$ are set as termination conditions. That is, when the number of iterations reaches $K_{\max}$ or the running time exceeds $T_{\max}$, the iteration stops and the current optimal path is output.

4. Simulation and Experimental Validation

In this section, we first present the design of the AUV 3D autonomous exploration simulation platform and the setting of key experimental parameters. To verify the effectiveness and superiority of our modular framework and improved algorithms, we conduct separate comparative experiments on path planning and trajectory optimization and tracking. We analyze the planning success rate, path quality, planning efficiency, and tracking accuracy of different algorithms under varying working conditions. We then carry out an integrated validation of the full framework in the 3D autonomous exploration scenario, evaluating the system's exploration coverage and operational efficiency in unknown underwater environments. Experimental results are presented with both quantitative data and visual visualization. We also conduct a detailed comparative analysis with traditional mainstream algorithms to demonstrate the advantages of our method in navigation safety, motion stability, and exploration efficiency.

4.1. Simulation Experiment Design

To accurately replicate the operational characteristics of AUVs in complex underwater environments, we select the BlueROV2 as the research subject [38]. Centered on the Software-In-The-Loop

 doi:10.20944/preprints202604.1168.v1

(SITL) mode of the ArduSub autopilot software stack, our system integrates the Gazebo 3D simulation environment and the Robot Operating System (ROS) communication framework. We achieve seamless integration between ROS control commands and the ArduSub flight control system through the MAVROS communication middleware. This constructs a complete velocity-attitude closed-loop control system. This system enables the full-process development and validation of AUVs path planning, trajectory optimization and tracking, and autonomous exploration modules. The overall system architecture adopts a layered design, consisting of four layers: the simulation layer, the perception and mapping layer, the exploration decision and planning layer, and the human-machine interaction layer. The corresponding architecture diagram is shown in Figure 7.

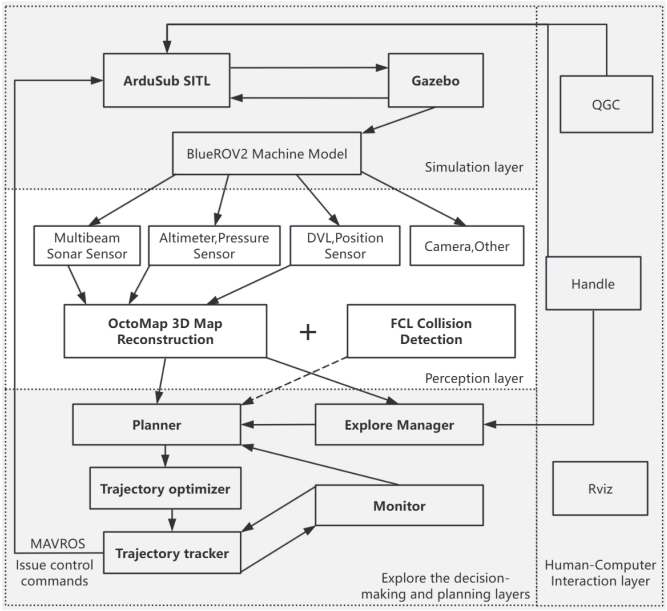
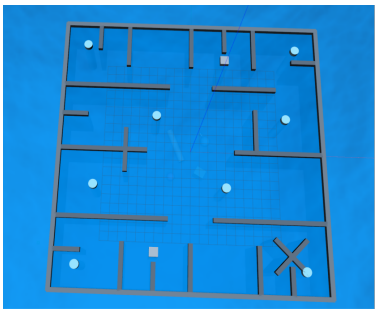
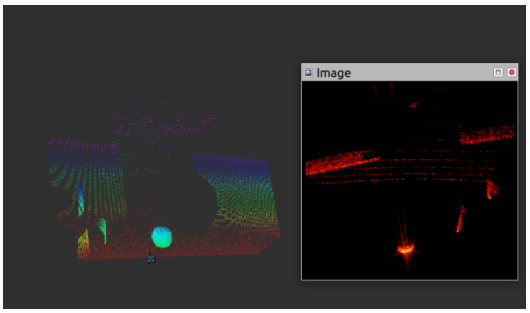


Figure 7. Overall system framework.

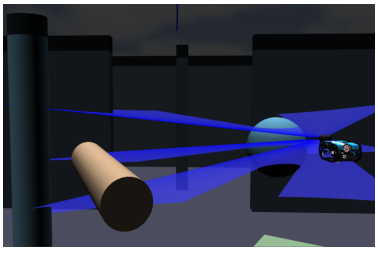
To validate the dynamic path planning and tracking obstacle avoidance control algorithms, as well as the 3D autonomous exploration algorithm, an underwater environment with dimensions of (30, 30, 6) meters is established in the Gazebo underwater environment for testing, as shown in Figure 8.



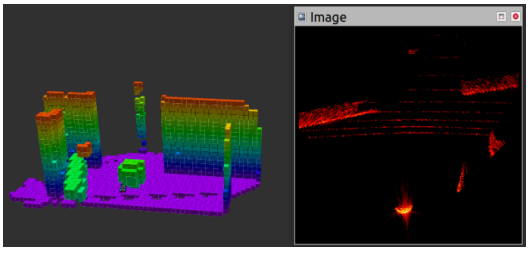
(**a**) Underwater World



(**b**) Sonar Point Cloud



(**c**) Underwater Vehicle



(**d**) OctoMap 3D Map

Figure 8. Comparison of the improved algorithm and the traditional algorithm.

 doi:10.20944/preprints202604.1168.v1

4.2. Simulation Results and Analysis

4.2.1. Path Planning Verification

In the simulated 3D static obstacle scenario, we set the start and goal points to $(2.06, -0.23, -2.50)$ and $(-0.11, 12.84, -2.82)$, respectively. The path planning performance of three algorithms under different obstacle avoidance weight parameters is compared.

As shown in the search trees of Figure 9, Figure 10, and Figure 11, all planners adopt a horizontal maneuvering strategy to circumvent obstacles when the obstacle avoidance weight is set to zero. Increasing this weight progressively expands the search volume along the z-axis for each planner, enabling exploration of a broader 3D configuration space and generating higher-quality paths. In practical underwater environments, this parameter can be tuned according to terrain complexity.

The data in Table 1, Table 2, and Table 3 further demonstrate that the A-IRRT* and I-LazyTheta* algorithms outperform conventional RRT* in both path quality and success rate. Due to inherent sampling randomness, Informed-RRT* produces varying paths across iterations. This results in higher average computation times compared to LazyTheta*. The collision caching mechanism in LazyTheta* reuses historical collision data when start/end points undergo minor adjustments, substantially improving computational efficiency. Experimental results confirm that Informed-RRT* achieves superior path quality in static environments. In contrast, LazyTheta* delivers faster response times for real-time applications with stringent latency constraints.

Table 1. Comparison of Planner Results With Obstacle Avoidance Weight of 0.0.

Planner Type	Weight	Success Rate	Path Length (m)	Planning Time (ms)
RRT*	0.0	88.7%	18.71	213
A-IRRT*	0.0	90.2%	14.52	85
I-LazyTheta*	0.0	89.3%	14.58	10

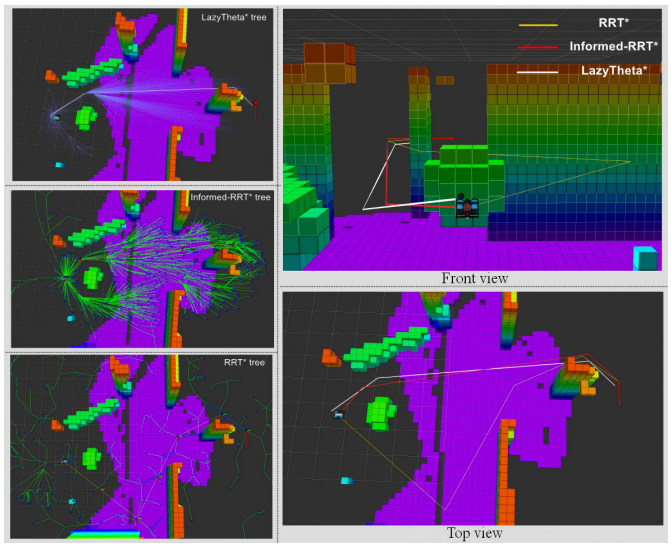


Figure 9. Visualization of each planner when the weight is 0.0.

Table 2. Comparison of Planner Results With Obstacle Avoidance Weight of 0.5.

Planner Type	Weight	Success Rate	Path Length (m)	Planning Time (ms)
RRT*	0.5	92.2%	15.24	222
A-IRRT*	0.5	96.8%	14.50	145
I-LazyTheta*	0.5	95.5%	14.11	4

 doi:10.20944/preprints202604.1168.v1

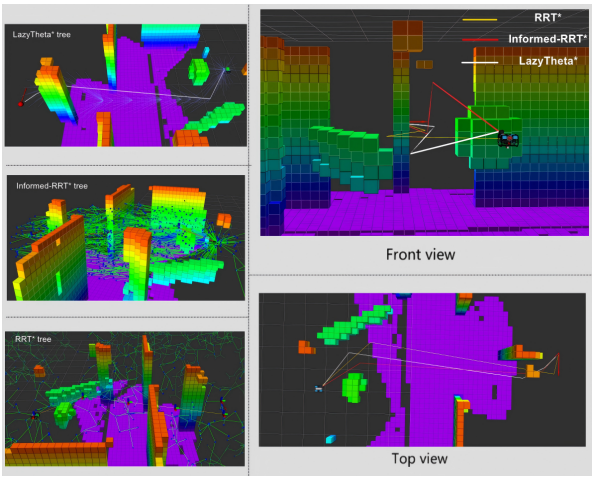


Figure 10. Visualization of each planner when the weight is 0.5.

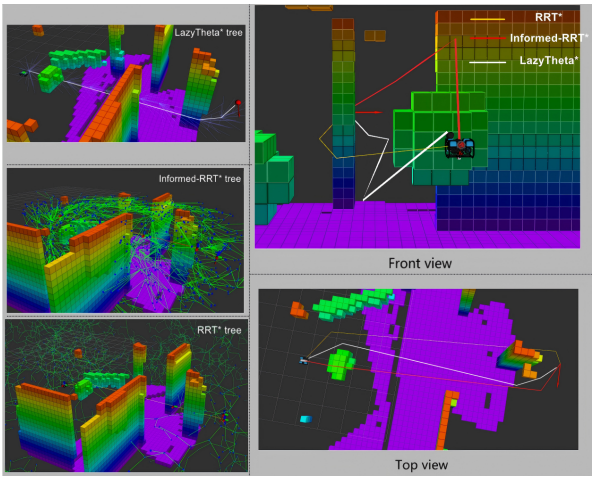


Figure 11. Visualization of each planner when the weight is 1.0.

Table 3. Comparison of Planner Results With Obstacle Avoidance Weight of 1.0.

Planner Type	Weight	Success Rate	Path Length (m)	Planning Time (ms)
RRT*	1.0	94.5%	15.24	225
A-IRRT*	1.0	96.4%	14.80	115
I-LazyTheta*	1.0	91.3%	14.08	4

4.2.2. Trajectory Optimization and Trajectory Tracking

After global path planning is completed, the resulting path typically consists of discrete path points. To meet the continuity requirements of trajectory tracking control, we smooth the discrete path using a trajectory optimizer to generate a parametric continuous trajectory. For the discrete path generated by the LazyTheta* algorithm, the effects of the two trajectory optimizers used in this design are shown in Figure 12.

Figure 12 illustrates that the trajectory generated by the B-spline curve exhibits superior smoothness but tends to be closer to obstacles. In contrast, the trajectory produced by polynomial fitting maintains a larger distance from obstacles, demonstrating higher safety.

 doi:10.20944/preprints202604.1168.v1

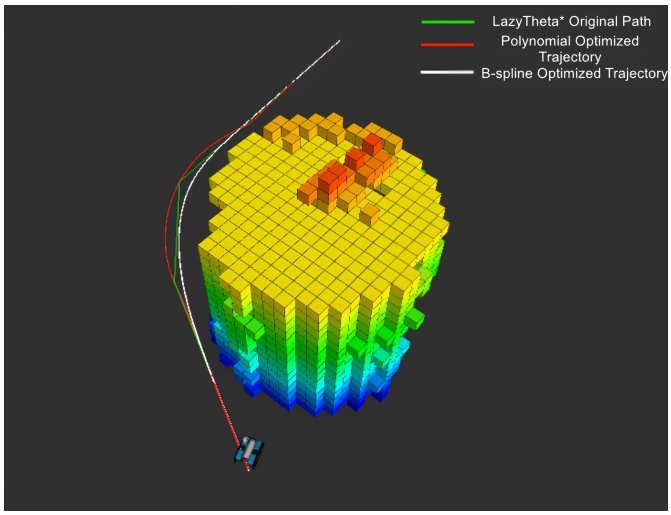


Figure 12. Trajectory optimization visualization diagram.

To validate the trajectory tracking performance of the trajectory optimization algorithm, this study constructed a test scenario in 3D space. We set the starting coordinates to$(-8.64, 9.12, 0.00)$, the goal coordinates to $(1.45, -4.25, -0.24)$, and the maximum velocity to 0.5 m/s. Control commands are sent to the ArduSub underwater vehicle via the MAVROS communication node, with the output parameters including the three-dimensional linear velocities (v_x, v_y, v_z) and the yaw angle.

To further optimize the control performance of the system, we finely tune the PID controller parameters of the ArduSub underwater vehicle through the QGroundControl (QGC) ground station. The final PID controller parameter configuration is listed in Table 4. The feedback PID eliminates errors, the feedforward (FF) term improves response speed, and integral anti-windup prevents saturation. This combination of parameters ensures control system stability while effectively enhancing the dynamic response performance.

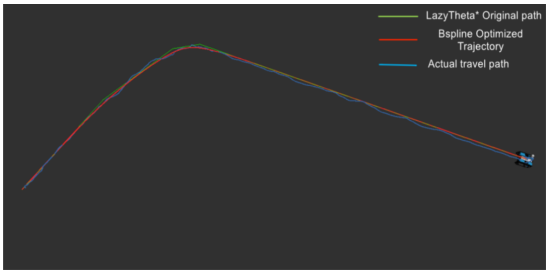
Table 4. ArduSub PID Parameters.

Parameter	ArduSub Parameter Name	Value
Proportional	PSC_VELXY_P	1.60
Integral	PSC_VELXY_I	0.60
Derivative	PSC_VELXY_D	0.27
Feedforward	PSC_VELXY_FF	3.00
Integral Limit	PSC_VELXY_IMAX	1000

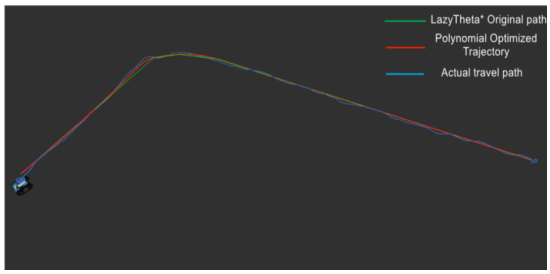
Comparing resultant velocity curves in Figure 13, the optimized actual velocity curve closely matches the desired value, with velocity fluctuations controlled within ±0.15 m/s. The mean heading angle control error is less than 1.2°. This verifies the closed-loop control system's precise tracking capability and meets obstacle avoidance tracking requirements in complex environments.

Combining the data analysis in Figure 13 and Table 5, we observe that polynomial trajectory optimization explicitly incorporates obstacle constraints into the cost function. It generates velocity curves with continuous curvature and significantly suppresses acceleration jumps, resulting in shorter tracking convergence time. However, this strategy incurs a high computational cost and a longer single optimization duration. Therefore, it is suitable for static environments with relaxed real-time requirements. B-spline optimization targets geometric smoothness of the trajectory without explicit collision detection, offering higher computational efficiency. Nevertheless, its velocity planning is significantly constrained by path curvature. This leads to noticeable velocity reduction in high-curvature regions and extended tracking time.

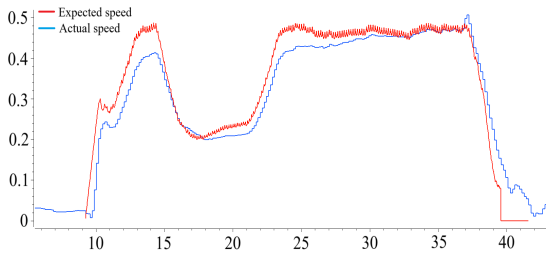
 doi:10.20944/preprints202604.1168.v1



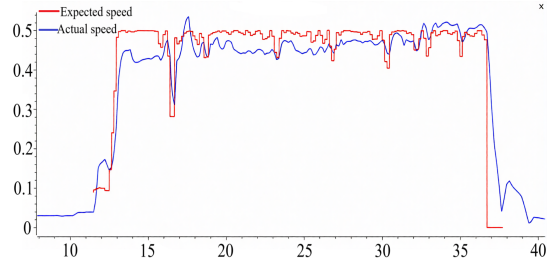
(**a**) Bspline Trajectory Optimization and Tracking



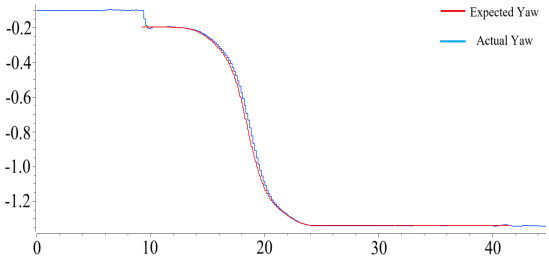
(**b**) Polynomial Trajectory Optimization and Tracking



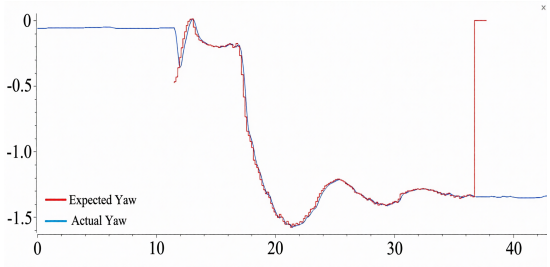
(**c**) Bspline Trajectory Tracking-Combined Velocity Curve



(**d**) Polynomial Trajectory Tracking-Combined Velocity Curve



(**e**) Bspline Trajectory Tracking-Yaw Angle Curve



(**f**) Polynomial Trajectory Tracking-Yaw Angle Curve

Figure 13. Comparison of trajectory optimization and tracking.

Table 5. Comparison of Trajectory Optimization and Tracking.

Optimization Method	Optimization Time (ms)	Planned Path Length (m)	Actual Tracked Length (m)	Avg. Velocity (m/s)	Tracking Time (s)
Polynomial Optimization	418	19.70	21.26	0.49	43.26
B-spline Optimization	51	18.92	20.15	0.43	46.66

Therefore, polynomial optimization is recommended for mission segments with dense obstacles and a priority on safety margins. B-spline optimization is recommended for open waters or maneuvering phases with stringent real-time constraints to achieve a trade-off between tracking performance and computational resources.

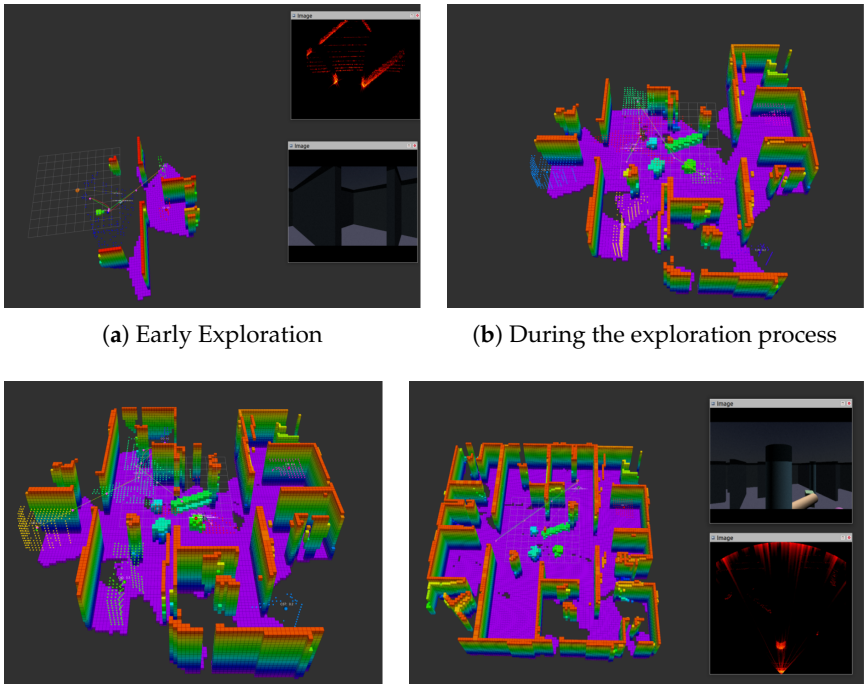
4.2.3. 3D Autonomous Exploration

After launching the simulation experiment program, we trigger the autonomous exploration mode via an external game controller. The vehicle immediately starts executing the exploration task in unknown regions. In the established OctoMap voxel-based map, the system extracts candidate frontier points at the boundaries between unknown and known areas using a frontier detection algorithm. These candidate points are then clustered into multiple frontier clusters (each cluster is visually distinguished by different colors), as shown in Figure 14. To optimize exploration efficiency,

 doi:10.20944/preprints202604.1168.v1

we select the geometric center of each cluster as the guiding target point. We then invoke the global path planner to generate a collision-free path, driving the vehicle toward the target region.

The entire exploration process from initiation to termination takes 17 minutes. During this process, some frontier points are actively filtered out. This occurs either because they are located in inaccessible areas for the vehicle (such as narrow passages or behind obstacles) or are deemed low-value by the information gain evaluation mechanism. Ultimately, the exploration coverage reaches 79.08%. The constructed map is generally consistent with the preset scenario of the simulation environment, satisfying the engineering application requirements for underwater vehicle exploration in unknown open-water environments.



(**a**) Early Exploration (**b**) During the exploration process

(**c**) During the exploration process (**d**) Exploration Complete

Figure 14. Comparison of the improved algorithm and the traditional algorithm.

5. Conclusions

In this paper, we propose a modular framework for AUV 3D autonomous exploration and trajectory tracking in unknown underwater environments. It addresses the core requirements of navigation safety, trackability, and exploration efficiency in complex 3D scenarios. The framework integrates three optimized core modules. The improved I-LazyTheta* and A-IRRT* algorithms generate collision-free paths efficiently with safety margin and obstacle avoidance weighting mechanisms. The trajectory tracking module achieves high-precision following via curvature-adaptive optimization. The 3D exploration strategy enables efficient unknown area traversal through multi-dimensional evaluation and TSP optimization. Simulation experiments verify the framework's effectiveness and superiority. It outperforms traditional methods in planning success rate, tracking accuracy and exploration coverage, providing reliable technical support for AUV autonomous navigation in unknown underwater environments.

In future work, we will incorporate actual underwater environmental interference models such as ocean currents and sonar noise to enhance the framework's robustness in real marine scenarios. We also plan to realize adaptive adjustment of key algorithm weights via intelligent optimization methods. Furthermore, we will expand our research to multi-AUV collaborative exploration to design efficient path planning and scheduling strategies for large-scale complex underwater areas.

 doi:10.20944/preprints202604.1168.v1

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

Funding: This work was supported in part by the Guangdong Key Discipline Research Capacity Building Project under Grant 2024ZDJS055; in part by University Research Project of Guangzhou Municipal Education Bureau under Grant 2024312146; in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2025A1515012983; in part by the Guangdong Characteristic Innovation Project for General Colleges and Universities under Grant 2025KTSCX107.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The dataset is available on request from the authors.

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

References

1. Song, B.; Pan, G.; Zhang, L.; et al. Development trend and key technologies of autonomous underwater vehicles. *Chinese Journal of Ship Research* **2022**, *17*, 27–44. https://doi.org/10.19693/j.issn.1673-3185.02939.
2. Sahoo, A.; Dwivedy, S.K.; Robi, P.S. Advancements in the field of autonomous underwater vehicle. *Ocean Engineering* **2019**, *181*, 145–160. https://doi.org/10.1016/j.oceaneng.2019.04.011.
3. Yan, Z.; Li, J.; Zhang, G.; Wu, Y. A Real-Time Reaction Obstacle Avoidance Algorithm for Autonomous Underwater Vehicles in Unknown Environments. *Sensors* **2018**, *18*, 438. https://doi.org/10.3390/s18020438.
4. Wang, C.; Yu, W.; Zhu, S.; Song, L.; Guan, X. Safety-Critical Trajectory Generation and Tracking Control of Autonomous Underwater Vehicles. *IEEE Journal of Oceanic Engineering* **2023**, *48*, 93–111. https://doi.org/10.1109/JOE.2022.3190635.
5. Yu, Y.; Guo, C.; Li, T.; Shen, H. Heading and velocity guidance based path following of autonomous surface vehicle with uncertainty attenuation and asymmetric saturated constraints. *ISA Transactions* **2023**, *138*, 88–105. https://doi.org/10.1016/j.isatra.2023.02.005.
6. Gan, W.; Zhu, D.; Hu, Z.; Shi, X.; Yang, L.; Chen, Y. Model predictive adaptive constraint tracking control for underwater vehicles. *IEEE Transactions on Industrial Electronics* **2020**, *67*, 7829–7840. https://doi.org/10.1109/TIE.2019.2941132.
7. Huang, H.; Zhang, S.; Fan, H.; et al. Real-time autonomous underwater and aerial exploration with limited FOV sensors. *J Real-Time Image Proc* **2025**, *22*, 118. https://doi.org/10.1007/s11554-025-01694-y.
8. Palomeras, N.; Hurtós, N.; Vidal, E.; Carreras, M. Autonomous exploration of complex underwater environments using a probabilistic next-best-view planner. *IEEE Robotics and Automation Letters* **2019**, *4*, 1619–1625. https://doi.org/10.1109/LRA.2019.2896759.
9. Dong, L.; Gan, X.; Li, H. Global-local hierarchical path planning method for unmanned surface vehicles based on dynamic constraints. *Journal of Marine Science and Technology* **2025**, *30*, 507–527. https://doi.org/10.1007/s00773-025-01070-2.
10. Devaurs, D.; Siméon, T.; Cortés, J. Optimal path planning in complex cost spaces with sampling-based algorithms. *IEEE Transactions on Automation Science and Engineering* **2016**, *13*, 415–424. https://doi.org/10.1109/TASE.2015.2487881.
11. Jiang, X.; Wang, Z.; Dong, C. A path planning algorithm based on improved RRT sampling region. *Comput. Mater. Contin.* **2024**, *80*, 4303–4323. https://doi.org/10.32604/cmc.2024.054640.
12. Ying, Y.; Li, Z.; Ruihong, G.; Yisa, H.; Haiyan, T.; Junxi, M. Path planning of mobile robot based on improved RRT algorithm. In *Proceedings of the 2019 Chinese Automation Congress (CAC)*; 2019; pp. 4741–4746. https://doi.org/10.1109/CAC48633.2019.8996415.
13. Li, J.; Li, C.; Chen, T.; Zhang, Y. Improved RRT algorithm for AUV target search in unknown 3D environment. *Journal of Marine Science and Engineering* **2022**, *10*, 826. https://doi.org/10.3390/jmse10060826.
14. Mendonca, P.; Goodwin, S. C-Theta*: Cluster based path-planning on grids. In *Proceedings of the 2015 International Conference on Computational Science and Computational Intelligence (CSCI)*; 2015; pp. 605–608. https://doi.org/10.1109/CSCI.2015.92.

 doi:10.20944/preprints202604.1168.v1

15. Khalidi, D.; Gujarathi, D.; Saha, I. T: A heuristic search based path planning algorithm for temporal logic specifications. In *Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA)*; 2020; pp. 8476–8482. https://doi.org/10.1109/ICRA40945.2020.9196928.
16. Aine, S.; Swaminathan, S.; Narayanan, V.; et al. Multi-Heuristic A*. *The International Journal of Robotics Research* **2016**, *35*, 224–243. https://doi.org/10.1177/0278364915594029.
17. Faria, M.; Marín, R.; Popović, M.; Maza, I.; Viguria, A. Efficient lazy theta* path planning over a sparse grid to explore large 3D volumes with a multirotor UAV. *Sensors* **2019**, *19*, 174. https://doi.org/10.3390/s19010174.
18. Chen, Z.; Yan, J.; Huang, R.; Gao, Y.; Peng, X.; Yuan, W. Path planning for autonomous underwater vehicles (AUVs) considering the influences and constraints of ocean currents. *Drones* **2024**, *8*, 348. https://doi.org/10.3390/drones8080348.
19. Li, K.; Li, L.; Tang, C.; Lu, W.; Fan, X. Three-dimensional path planning based on six-direction search scheme. *Sensors* **2024**, *24*, 1193. https://doi.org/10.3390/s24041193.
20. Tran, N.H.; Nguyen, A.D.; Nguyen, T.N. A genetic algorithm application in planning path using B-spline model for autonomous underwater vehicle (AUV). *Applied Mechanics and Materials* **2020**, *902*, 54–64. https://doi.org/10.4028/www.scientific.net/amm.902.54.
21. Nguyen, N.T.; Schilling, L.; Angern, M.S.; Hamann, H.; Ernst, F.; Schildbach, G. B-spline path planner for safe navigation of mobile robots. In *Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*; 2021; pp. 339–345. https://doi.org/10.1109/IROS51168.2021.9636612.
22. Sun, W.; Sun, P.; Ding, W.; Zhao, J.; Li, Y. Gradient-based autonomous obstacle avoidance trajectory planning for B-spline UAVs. *Scientific Reports* **2024**, *14*, 14458. https://doi.org/10.1038/s41598-024-65463-w.
23. Topiwala, A.; Inani, P.; Kathpal, A. Frontier based exploration for autonomous robot. *arXiv preprint* **2018**, arXiv:1806.03581. https://doi.org/10.48550/arXiv.1806.03581.
24. Vidal, E.; Palomeras, N.; Istenič, K.; Hernández, J.D.; Carreras, M. Two-dimensional frontier-based viewpoint generation for exploring and mapping underwater environments. *Sensors* **2019**, *19*, 1460. https://doi.org/10.3390/s19061460.
25. Xiang, D.; Lin, H.; Ouyang, J.; et al. Combined improved A* and greedy algorithm for path planning of multi-objective mobile robot. *Scientific Reports* **2022**, *12*, 13273. https://doi.org/10.1038/s41598-022-17684-0.
26. Scharff Willners, J.; Gonzalez-Adell, D.; Hernández, J.D.; Pairet, È.; Petillot, Y. Online 3-Dimensional Path Planning with Kinematic Constraints in Unknown Environments Using Hybrid A* with Tree Pruning. *Sensors* **2021**, *21*, 1152. https://doi.org/10.3390/s21041152.
27. Nash, A.; Koenig, S.; Tovey, C.A. Lazy Theta*: Any-angle path planning and path length analysis in 3D. In *Proceedings of the Third Annual Symposium on Combinatorial Search (SOCS 2010)*; 2010; pp. 153–154. https://doi.org/10.1609/SOCS.V1I1.18152.
28. Noreen, I.; Khan, A.; Habib, Z. Optimal path planning using RRT* based approaches: A survey and future directions. *International Journal of Advanced Computer Science and Applications* **2016**, *7*. https://doi.org/10.14569/IJACSA.2016.071114.
29. Kim, M.-C.; Song, J.-B. Informed RRT* towards optimality by reducing size of hyperellipsoid. In *Proceedings of the 2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*; 2015; pp. 244–248. https://doi.org/10.1109/AIM.2015.7222539.
30. Jia, T.; et al. OMU: A probabilistic 3D occupancy mapping accelerator for real-time OctoMap at the edge. In *Proceedings of the 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*; 2022; pp. 909–914. https://doi.org/10.23919/DATE54114.2022.9774508.
31. Pan, J.; Chitta, S.; Manocha, D. FCL: A general purpose library for collision and proximity queries. In *Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA)*; 2012; pp. 3859–3866. https://doi.org/10.1109/ICRA.2012.6225337.
32. Velikzhanin, A.; Skarga-Bandurova, I. A Bresenham-based global path planning algorithm on grid maps. In *Proceedings of the 2023 13th International Conference on Dependable Systems, Services and Technologies (DESSERT)*; 2023; pp. 1–8. https://doi.org/10.1109/DESSERT61349.2023.10416444.
33. Ruan, X.; Chen, X.; Zhu, X. Mobile robot exploration strategy based on multiple information gain. *Journal of Beijing University of Technology* **2023**, *49*, 990–998. https://doi.org/10.11936/bjutxb2021120005.
34. Selin, M.; Tiger, M.; Duberg, D.; Heintz, F.; Jensfelt, P. Efficient autonomous exploration planning of large-scale 3-D environments. *IEEE Robotics and Automation Letters* **2019**, *4*, 1699–1706. https://doi.org/10.1109/LRA.2019.2897343.

 doi:10.20944/preprints202604.1168.v1

35. Kulkarni, O.; Burhanpurwala, A. A survey of advancements in DBSCAN clustering algorithms for big data. In *Proceedings of the 2024 3rd International Conference on Power Electronics and IoT Applications in Renewable Energy and its Control (PARC)*; 2024; pp. 106–111. https://doi.org/10.1109/PARC59193.2024.10486339.
36. Peng, J.; Fan, J. Solving TSP problem using reinforcement learning algorithm integrated with 2-opt. *Computer Science* **2025**, *52*, 182–189.
37. Huang, Q.; Feng, Z.; Du, Y.; et al. Research on greedy path optimization algorithm based on K-means and 2-Opt improvement. *Automation and Information Engineering* **2025**, *46*, 9–17.
38. Blue Robotics. BlueROV2: The world's most affordable high-performance ROV. BlueROV2 Datasheet, 2016; pp. 1–6.