Journal of
Marine Science and Engineering

MDPI

Article

A Trajectory Tracking and Local Path Planning Control Strategy for Unmanned Underwater Vehicles

Xun Zhang [1,2,3], Ziqi Wang [1,2,*], Huijun Chen [1,2] and Hao Ding [1,2]

[1] College of Intelligent Systems Science and Engineering, Harbin Engineering University, Harbin 150001, China; zhangxun2008@hrbeu.edu.cn (X.Z.); chenhuijun@hrbeu.edu.cn (H.C.); s322520216@hrbeu.edu.cn (H.D.)
[2] Qingdao Innovation and Development Center, Harbin Engineering University, Qingdao 266000, China
[3] Sanya Nanhai Innovation and Development Base of Harbin Engineering University, Sanya 572024, China
* Correspondence: wzq_chxym@hrbeu.edu.cn

Abstract: The control strategy of an underdriven unmanned underwater vehicle (UUV) equipped with front sonar and actuator faults in a continuous task environment is investigated. Considering trajectory tracking and local path planning in complex-obstacle environments, we propose a task transition strategy under the event-triggered mechanism and design the corresponding state space and action space for the trajectory tracking task under the deep reinforcement learning framework. Meanwhile, a feed-forward compensation mechanism is designed to counteract the effects of external disturbances and actuator faults in combination with a reduced-order extended state observer. For the path planning task under the rapidly exploring random tree (RRT) framework, a reward component and angular factors are introduced to optimize the growth and exploration points of the extended tree under the consideration of the shortest distance, optimal energy consumption, and steering angle constraints. The effectiveness of the proposed method was verified through continuous task simulations of trajectory tracking and local path planning.

Keywords: deep reinforcement learning; RRT; unmanned underwater vehicle; path tracking; route planning

check for updates

Citation: Zhang, X.; Wang, Z.; Chen, H.; Ding, H. A Trajectory Tracking and Local Path Planning Control Strategy for Unmanned Underwater Vehicles. *J. Mar. Sci. Eng.* **2023**, *11*, 2230. https://doi.org/10.3390/jmse11122230

Academic Editor: Eugen Rusu

Received: 1 November 2023
Revised: 18 November 2023
Accepted: 21 November 2023
Published: 25 November 2023

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

1. Introduction

With the emergence of unmanned underwater vehicles (UUVs) demonstrating high maneuverability, stealth, and intelligence in the tasks of detecting underwater pipelines, surveying underwater topography, exploring oil and gas resources, and undertaking military surveillance and patrolling activities, their control strategies in harsh ocean environments with poor communication conditions and low visibility have become a hot research topic for many scholars [1]. Since the underwater trajectory tracking and local path planning of a single UUV are the basis and prerequisite for accomplishing various complex tasks, granting UUVs these two capabilities simultaneously in complex underwater environments has become a top priority.

There have been numerous attempts at trajectory tracking with classical control methods. A fault-tolerant controller using feedback-linearized control to solve the consensus control problem for multi-UV systems was presented in [2]. Additionally, in [3], the target tracking control issue of underactuated UUVs was effectively tackled through the implementation of line-of-sight (LOS) and backstepping methods. In [4], a novel Lyapunov-based MPC framework was proposed to improve UUV tracking performance, and trajectory tracking control was explored using the model predictive control (MPC) approach. In order to attenuate the effect of the accuracy of the UUV mathematical model on the actual UUV control, many scholars have made improvements to the robustness and control algorithms of UUVs. In [5], an improved self-resistant (ADRC) control strategy was proposed, which estimated unknown disturbances by designing a generalized expanded state observer (ESO). Also, in terms of resistance to underwater disturbances, the motion control of an

overactuated UUV based on dynamic sliding mode control (DSMC) theory was presented in [6] to improve the system robustness under the effects of the ocean current and model uncertainties. On this basis, in [7], an optimal allocation control (AC) algorithm was designed to maintain the linear position and Euler angles of the UUV in the presence of model uncertainties as well as ocean currents and to minimize the energy consumption of the system. To solve the UUV bottom-following control accuracy problem, in [8], a model-free controller based on a fuzzy approximator was provided, which significantly improved the robustness and control accuracy. Although the proposals in [4,8] had significant advantages in terms of control accuracy, their handling of external perturbations was poorly considered in comparison to the solutions in [5,7], resulting in possible poor adaptation to real environments. In contrast to the traditional methods described above, reinforcement learning learns directly by interacting with the environment, and the control accuracy does not depend on the accuracy of the model [9,10].

Introducing reinforcement learning into the field of UUV control is an exciting direction. In [11], a deep deterministic policy gradient (DDPG)-based algorithm was implemented to control the velocity and attitude angle of a UUV; similarly, in [12], the DDPG algorithm was used to control the isotropic linear motion of a three-degrees-of-freedom UUV. At a high level of motion control, in [13], the trajectory tracking control task of a UUV was decomposed into multiple position tracking tasks, and the DDPG algorithm was used to realize the 3D trajectory tracking control of the UUV. In [14], pictures were innovatively used as inputs to the policy network, and the proximal policy optimization algorithm was employed to train the UUV to track underwater pipes. However, the above methods tended to ignore the actuator failure and perturbation states of UUVs, and the simplified few-degrees-of-freedom UUV model is simpler to control; thus, the application scenarios are limited.

In terms of obstacle avoidance and path planning, the mainstream methods can be roughly divided into three categories: environment modeling, traditional search, and meta-heuristic search. Among them, meta-heuristic search algorithms have become the current research hotspot in this direction due to their powerful exploration and adaptation capabilities. Environmental modeling methods include visual graphs, Voronoi diagrams, and grid-based techniques, while traditional search methods are mainly used for local path planning, including artificial potential field methods, dynamic window methods, and rapidly exploring random tree methods [15]. For global path planning, the rise of meta-heuristic algorithms has led to much more efficient and successful planning. These methods include the water wave optimization algorithm (WWO) [16], gray wolf optimization algorithm (GWO) [17], artificial fish swarm algorithm (AFSA) [18], and whale optimization algorithm (WOA) [19]. However, these algorithms have fewer applications in real UUV mission environments and are difficult to integrate with trajectory tracking tasks in the temporal and spatial domains. On the contrary, the RRT algorithm is widely used in global path planning and local replanning due to its strong environmental adaptability and search efficiency.

In [20], an improved RRT algorithm was presented to solve the problem of target search and interception in an unknown three-dimensional (3D) environment. In [21], an improved AAF-RRT algorithm that combined the artificial potential field method (APF) with the RRT algorithm was proposed, and a two-layer path planner that could pass through narrow channels was designed. Additionally, in [22], to tackle the smoothness problem of the paths planned by RRT, the planned paths were smoothed by combining greedy algorithms in order to satisfy the special requirements of UUVs for the shortest distance and maneuverability. The methods in both [21,22] were optimized by optimizing the process of generating new path nodes using the RRT algorithm, which is the current direction of RRT algorithm development. The difference is that the combination with the APF algorithm directly changed the position of the newly generated node, while the greedy algorithm was optimized in terms of path smoothness. The problem with the greedy

algorithm is that the planned paths tend to lose some of their optimal characteristics, which is why we chose to incorporate angular constraints.

In our task environment, the advantage of RRT is that its search approach is based on a tree structure with random sampling, and routes from points to be extended with new points can generate kinematics and dynamics simulations of the robot, which makes its combination with the tracking task in the same temporal and spatial domains more efficient [21]. Secondly, the search process of RRT is more concise compared to other meta-heuristic algorithms and does not require precise spatial information about the obstacles.

The main contributions of our work are summarized below:

- A trajectory tracking controller is designed by using the event triggering mechanism and combining the fault state and disturbance state of the UUV actuator, which realizes the classification and collection of the relevant states in the model training and simulation and the seamless connection of the successive tasks.
- Combining the deep reinforcement learning framework with meta-heuristic algorithms in the same task space, the conditions of distance, energy consumption, and steering angle constraints required by UUVs in the task environment are considered from various perspectives, and reward component and angular factors are introduced into the RRT algorithm to locally plan the path.
- In response to the effects of external perturbations and actuator faults, a joint reduced-order extended state observer and feed-forward compensation mechanism is designed to enable the UUV to counteract the unfavorable effects by automatically adjusting its own output.

2. Preliminaries

In this section, we will model the mission environment, starting from a mathematical model of an underactuated UUV with thruster faults, and combine our knowledge of deep reinforcement learning and RRT to model the environment for the tracking task and local planning task.

2.1. Underactuated UUV Model Description

A torpedo-shaped underactuated UUV, which employs a cross-rudder and single-propeller actuator layout, was chosen as the research object of this paper [23]. We define the position, attitude, and velocity of the underdriven UUV in the Earth-fixed coordinate system $(O_E - X_E Y_E Z_E)$ and body-axis coordinate system $(O_B - X_B Y_B Z_B)$, respectively. As shown in Figure 1, (x, y, z) denote the position of the UUV in the Earth-fixed coordinate system. θ and φ represent the pitch angle and yaw angle of the UUV in the Earth-fixed coordinate system, respectively, and their positive direction follows the right-hand rule. u, v, and w are the velocities in surge, sway, and heave, respectively. Finally, q and r refer to pitch angular velocity and yaw angular velocity in the body-axis coordinate system, respectively, and their positive direction is the same as that of θ and φ.

Based on the description of Fossen's equations [24] associated with UUV thruster failure [25], we can obtain the five-degrees-of-freedom UUV dynamics equations with unknown perturbations:

$$\begin{cases} \dot{u} = \frac{m_{22}}{m_{11}} vr - \frac{m_{33}}{m_{11}} wq - \frac{1}{m_{11}} f_u + \frac{1}{m_{11}} (\lambda_{uh} \tau_u + \bar{\omega}_{uh}) - \frac{1}{m_{11}} \tau_{du} \\ \dot{v} = -\frac{m_{11}}{m_{22}} ur - f_v - \frac{1}{m_{22}} \tau_{dv}(t) \\ \dot{w} = \frac{m_{11}}{m_{22}} uq - f_w - \frac{1}{m_{33}} \tau_{dv}(t) \\ \dot{q} = \frac{m_{33} - m_{11}}{m_{55}} uw - f_q - \frac{\rho_e g \nabla G M_L \sin(\theta)}{m_{55}} + \frac{1}{m_{55}} \left(\lambda_{qh} \tau_q + \bar{\omega}_{qh} \right) - \frac{1}{m_{55}} \tau_{dq} \\ \dot{r} = \frac{m_{11} - m_{22}}{m_{66}} uv - f_r + \frac{1}{m_{66}} (\lambda_{rh} \tau_r + \bar{\omega}_{rh}) - \frac{1}{m_{66}} \tau_{dr} \end{cases} \tag{1}$$

where m_{11}, m_{22}, m_{33}, m_{55}, and m_{66} represent the mass and inertia parameters of the UUV, and τ_u, τ_q, and τ_r represent the control force and moment generated by the tail propeller and rudder. f_u, f_v, f_w, f_q, and f_r represent hydrodynamic damping and friction items. $\tau_{du}(t)$, $\tau_{dv}(t)$, $\tau_{dw}(t)$, $\tau_{dq}(t)$, and $\tau_{dr}(t)$ represent time-varying bounded environmental disturbances caused by factors such as wind and ocean currents. GM_L, ∇, g, and ρ_e represent the longitudinal metacentric height, displacement, gravity acceleration, and water density, respectively.

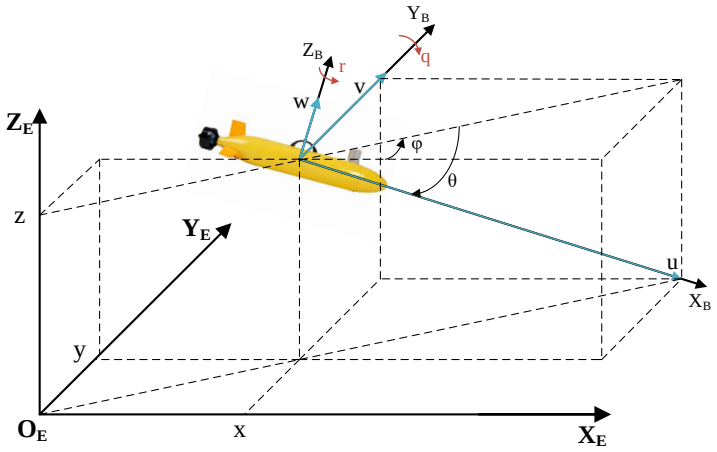


Figure 1. Coordinate system of underactuated UUV.

In order to more realistically simulate the impact of the UUV's thruster faults on its drive accuracy and drive efficiency in underwater environments, we simulate the loss of control signals for UUVs when they are subjected to rudder vibration and propeller entanglement by a combination of multiplicative and additive faults. In Equation (1), we set $\tau_i(i = u, q, r)$ as the input of the actuator; λ_{uh}, λ_{qh}, and λ_{rh} are set at a fixed value; and additive faults $\bar{\omega}_{uh}$, $\bar{\omega}_{qh}$, and $\bar{\omega}_{rh}$ are time-varying variables.

2.2. Tracking the State Model of the Mission Environment

In the tracking task, we select the environment state as the relative distance ρ, relative pitch angle β, and relative yaw angle α between the UUV and the desired trajectory in each time step, and obtain it by the LOS method [3]. As shown in Figure 2, (x_m, y_m, z_m) are the coordinates of the desired trajectory point in the Earth's fixed coordinate system, and $\Delta x = x_m - x$, $\Delta y = y_m - y$, and $\Delta z = z_m - z$ denote the three-dimensional coordinate difference of the trajectory point with respect to the UUV. (x_r, y_r, z_r) are the relative errors and can be expressed as

$$\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = R_1^T(\theta, \varphi) \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} \tag{2}$$

where R_1 can be expressed as

$$R_1(\theta, \varphi) = \begin{bmatrix} \cos(\theta)\cos(\varphi) & -\sin(\varphi) & \sin(\theta)\cos(\varphi) \\ \cos(\theta)\sin(\varphi) & \cos(\varphi) & \sin(\theta)\sin(\varphi) \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{3}$$

Finally, we can obtain the relative distance ρ, relative pitch angle β, and relative yaw angle α by

$$\begin{aligned} \rho &= \sqrt{x_r^2 + y_r^2 + z_r^2} \\ \beta &= \arctan 2(z_r, \sqrt{x_r^2 + y_r^2}) \\ \alpha &= \arctan 2(y_r, x_r) \end{aligned} \tag{4}$$

where arctan $2(a, b)$ indicates the inverse tangent value a/b.

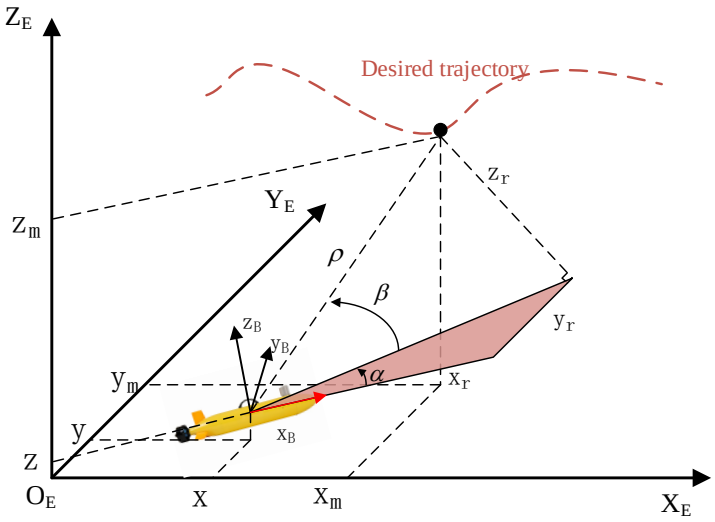


Figure 2. Schematic of the trajectory tracking task environment.

Remark 1. *Some UUV physical constraints and assumptions are as follows:*

- *The motion of the UUV in the roll direction is neglected.*
- *The UUV has neutral buoyancy, and the origin of the body-fixed coordinates is located at the center of mass.*
- *The yaw angle is limited to* $(-\pi, \pi]$*, and the pitch angle is limited to* $(-6/\pi, 6/\pi]$.

To achieve the ability to train UUVs to track a target through deep reinforcement learning, the policy network input states, output actions, and reward functions need to be formulated in conjunction with the states obtained from Equation (4) relative to the desired trajectory.

1. Trajectory Tracking Task Status
   We selected the velocity state, actuator failure state, and perturbation state, which are directly related to the update of the UUV position, velocity, and acceleration states, as the base states for the policy network inputs. The velocity state can be expressed as

$$g_s \doteq v = [u\ v\ w\ q\ r]^T \tag{5}$$

   Since multiplicative faults are approximated as fixed values, so that they have no effect on the learning update process of the policy network, the fault state should be set as additive faults, denoted as follows:

$$f_s \doteq [\,\bar{\omega}_{uh}\ \ \bar{\omega}_{qh}\ \ \bar{\omega}_{rh}]^T \tag{6}$$

   The five-degrees-of-freedom UUV perturbation state can be defined as

$$\boldsymbol{h}_s = \left[\ \tau_{du}\quad \tau_{dv}\quad \tau_{dw}\quad \tau_{dq}\quad \tau_{dr}\ \right] \tag{7}$$

   In addition, combining the relative distance, relative pitch angle, and relative yaw angle obtained from Equation (4), we set the error state of the trajectory tracking task as follows:

$$e_s = [\rho_d - \rho, \beta_d - \beta, \alpha_d - \alpha]^T \tag{8}$$

   where ρ_d, β_d, and α_d denote the desired relative distance, relative pitch angle, and relative error, respectively.

Finally, we can obtain the input state of the trajectory tracking task in the deep reinforcement learning framework:

$$s = [g_s \ f_s \ h_s \ e_s] \tag{9}$$

2. Action space
The UUV studied in this paper is powered by a propeller in the tail and rudders on both sides, with control forces and moments applied to the UUV to update its state over time. Therefore, we directly chose the control force and moment as the output of the deep reinforcement learning strategy network, which can be expressed as

$$a = [\tau_u, \tau_q, \tau_r]^T \tag{10}$$

3. Reward function
The trajectory tracking task expects the UUV to track the target quickly and maintain a fixed tracking state in a limited time, so the reward function should be selected by first considering the error between the UUV and the desired trajectory, based on the approximation that the larger the error, the smaller the reward. Combining this with Equation (8), one obtains

$$r_s = -\zeta_s \|e_s\|_1 \tag{11}$$

In addition, we expect the outputs of the UUV propeller and rudder to be maintained in a more stable and energy-efficient manner, which means that under the condition of achieving the same control accuracy, the smaller the output value, the better the corresponding action. Thus, we define the reward function related to the controller output as

$$r_p = -\zeta_p a^T a \tag{12}$$

where ζ_s and ζ_p denote the weight parameters of the two-part reward function.

2.3. Reduced-Order Extended State Observer

To deal with the unknown perturbations and actuator faults of underdriven UUVs, we design a reduced-order extended observer (RESO) to estimate the perturbation terms and additive fault terms of UUVs in real environments. First, we define the combination term of external perturbations and actuator additive faults as

$$\chi = \begin{bmatrix} \chi_u \\ \chi_v \\ \chi_w \\ \chi_q \\ \chi_r \end{bmatrix} = \begin{bmatrix} \tau_{du} \\ \tau_{dv} \\ \tau_{dw} \\ \tau_{dq} \\ \tau_{dr} \end{bmatrix} + \begin{bmatrix} \bar{\omega}_{uh} \\ \bar{\omega}_{vh} \\ \bar{\omega}_{wh} \\ \bar{\omega}_{qh} \\ \bar{\omega}_{rh} \end{bmatrix} \tag{13}$$

Assuming that the external perturbation and the actuator fault are both continuously minimizable, then χ is also continuously minimizable. We can rewrite Equation (1) in matrix form:

$$M\dot{v} + C(v)v + D(v)v + g(\eta) = \lambda a + \chi \tag{14}$$

where M, C, and D represent the inertia matrix, the Coriolis force, and the centrifugal terms matrix, respectively. The multiplicative faults λ of the three actuators are set to 0.8. $v = [u, v, w, q, r]^T$ represents the velocity vector, and $\eta = [x, y, z, \theta, \varphi]^T$ represents the position and attitude vectors. a denotes the same action space as in Equation (10). Then, this is reduced to the standard form of a reduced-dimensional observation:

$$\dot{v} = -M^{-1}(C(v)v + D(v)v + g(\eta)) + \alpha u + \alpha\chi \tag{15}$$

where u represents λa in Equation (14). For a detailed derivation of the dimensionality-

reducing dilation observer, please refer to [26]. Ultimately, the obtained reduced-dimensional extended observer is

$$\begin{cases} \delta = \dot{v} - v \\ \dot{\hat{v}} = -M^{-1}(C(v)v + D(v)v + g(\eta)) - \beta_1 f_1(\delta) \quad + \alpha(\hat{\chi} + u) \\ \dot{\hat{\chi}} = -\frac{\beta_2}{\chi} f_2(\delta) \end{cases} \tag{16}$$

Here, f_1 and f_2 denote the parametric functions, which can be expressed as

$$\begin{cases} f_1(\delta) = k_1 \delta \\ f_2(\delta) = k_2 |\delta|^{\partial} \mathrm{sgn}(\delta) \end{cases} \tag{17}$$

where k_1, k_2 and ∂ are set to constant values. For the proof of Lyapunov stability of this observer, please refer to Appendix A.

2.4. Localized Planning Mission Environment Model

To realize omnidirectional detection centered on the UUV, a 360-degree scanning sonar was selected to obtain scanning data [27]. As shown in Figure 3, an obstacle is considered detected if its relative distance to the UUV is less than or equal to $\rho_{\max}$.

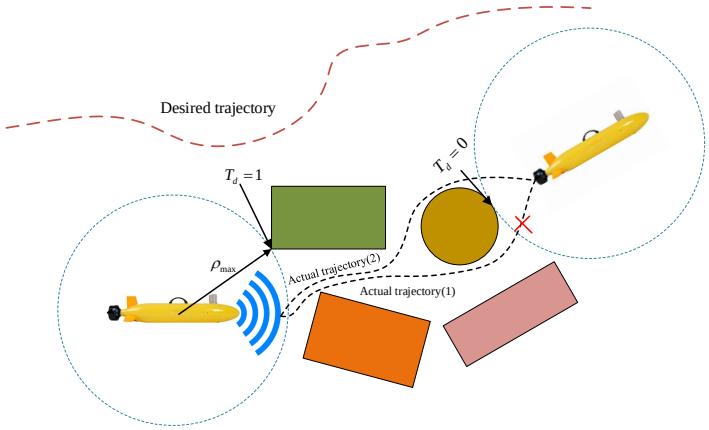


Figure 3. Schematic of the local path planning task environment.

At the same time, we introduce the event trigger flag T_d. When the UUV detects an obstacle, T_d becomes 1, indicating that the trajectory tracking task is temporarily stopped and the local planning task is started; when the UUV travels along the planned collision-free path to the point of the path where the obstacle is not detected, T_d turns back to 0, indicating that the local planning task is completed, and then the UUV continues to follow the desired trajectory.

Since the RRT-planned path is a time series consisting of the coordinates of the UUV corresponding to the time steps in the Earth's fixed coordinate system, the planned path can be represented by the following mathematical expression:

$$p(s) = \begin{cases} \bar{x}(s) \\ \bar{y}(s) \quad s \in [0, s_n] \\ \bar{z}(s) \end{cases} \tag{18}$$

where s denotes the path length corresponding to the unit time step, and s_n denotes the distance between the planning start and end points. At the same time, we introduce the concepts of the relative pitch angle and relative yaw angle from Equation (4) into the two neighboring nodes of the planning path and discretize them as

$$\begin{aligned} \bar{\beta}(n) &= \arctan 2\left(\bar{z}_r(n), \sqrt{\bar{x}_r(n) + \bar{y}_r(n)}\right) \\ \bar{\alpha}(n) &= \arctan 2(\bar{y}_r(n), \bar{x}_r(n)) \end{aligned} \tag{19}$$

2.5. The Principles of the Basic RRT Algorithm

The concept of the RRT algorithm can be briefly described as follows: First, the random tree T is initialized with the start node of the UUV; then, a random point is evenly sampled in the barrier-free space, and a nearby node in the random tree is traversed by the distance evaluation function. Secondly, a new node is generated by advancing with a unit step from the nearby node to a random node. If there is no collision between the nearby node and the new node, the new node is discarded without any modification to the random tree T [28]. The iterative process above is repeated until the target node becomes a leaf node, or the search ends when the set number of iterations is exceeded. Returning from the target node to the starting point, the planned path can be obtained. Figure 4 shows the node expansion process of basic RRT, as described above.

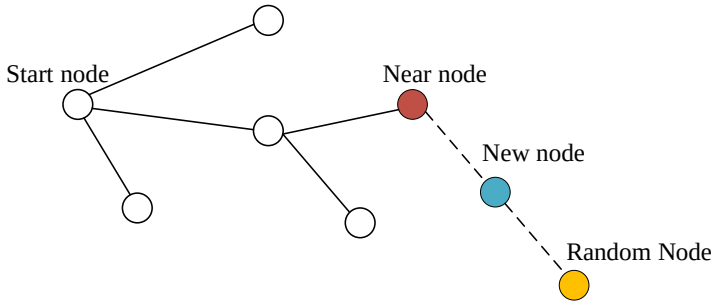


Figure 4. Schematic diagram of RRT algorithm.

3. Improved RRT Algorithm with Reinforcement Learning Applications

In this section, a steering angle optimization RRT algorithm based on a reward mechanism is proposed. The logic of deep reinforcement learning applied in this paper is also introduced. Finally, the structural framework of complete trajectory tracking combined with local planning is obtained.

3.1. Improved RRT Algorithm

In order to simulate the real underwater motion of the UUV, we limit the generation of new nodes in the RRT algorithm by using the maximum rotation angles $\theta_{\max}$ and $\varphi_{\max}$ of the UUV rotating around the y-axis and z-axis of the body-axis coordinate system as the constraints.

The starting point q_{rand} in the state space is used as the root node, and a point q_{rand} is randomly selected from the search space. If this point falls in the non-obstacle interval, it will traverse T to find the nearest node n to this random point [29]. Then, by applying Equation (19), we find the relative pitch angle $\bar{\beta}(s)$ and relative yaw angle $\bar{\alpha}(s)$ between q_{rand} and $q_{nearest}$. Finally, we calculate q_{new} under the condition of maximizing angle $\bar{\beta}_{\max}$ and $\bar{\alpha}_{\max}$:

$$q_{\text{new}} = \begin{cases} (q_{\text{rand}} - q_{\text{near}}) \cdot \bar{\beta}_{\max}, \bar{\beta}(n) > \bar{\beta}_{\max}, \bar{\alpha}(n) <= \bar{\alpha}_{\max} \\ (q_{\text{rand}} - q_{\text{near}}) \cdot \bar{\alpha}_{\max}, \bar{\alpha}(n) > \bar{\alpha}_{\max}, \bar{\beta}(n) <= \bar{\beta}_{\max} \\ (q_{\text{rand}} - q_{\text{near}}) \cdot \min(\bar{\beta}_{\max}, \bar{\alpha}_{\max}), \bar{\alpha}(n) > \bar{\alpha}_{\max}, \bar{\beta}(n) > \bar{\beta}_{\max} \end{cases} \tag{20}$$

In addition, for the case where $\bar{\beta}(n)$ and $\bar{\alpha}(n)$ are both within the angular constraints, the generation of their new nodes follows the rules of the basic RRT.

In fact, even though we limit the maximum steering angle of the planned path, we still cannot avoid the search instability and hysteresis caused by the strong randomness of the RRT algorithm. In this case, we introduce a reward function for the behavior of RRT to

generate new node q_{new} in order to reduce its unnecessary exploration. The setting of the reward weight should consider the following three aspects:

1. Rewarding the generation of new nodes that are close to the goal point q_{goal}.
   According to the basic RRT algorithm, the extension vector of the nearest point $q_{near}(n)$ can be expressed by the following mathematical formula [21]:

$$R(n) = \tau \times \frac{q_{rand}(n) - q_{near}(n)}{\|q_{rand}(n) - q_{near}(n)\|} \tag{21}$$

   where $R(n)$ denotes the expansion vector from the nth nearest point $q_{near}(n)$ to the nth random point $q_{rand}(n)$, and τ represents the step size in the RRT algorithm. Then, the reward component from q_{goal} to $q_{near}(n)$ can be defined as

$$G_1(n) = g_1 \times \tau \times \frac{q_{goal} - q_{near}(n)}{\left\|q_{goal} - q_{near}(n)\right\|} \tag{22}$$

   where g_1 denotes the reward gain coefficient that varies the target bias of the generated random vector.
2. Rewarding the generation of new nodes that are close to the desired trajectory.
   The continuous task environment of trajectory tracking and localized obstacle avoidance requires the UUV to complete the switch between the two task states with the smallest possible energy consumption. Therefore, it requires us to keep the error with the desired trajectory small during the planning of the paths. As shown in Figure 3, we plan two actual paths, and only that which is closer to the desired trajectory is selected. Emulating the form of the reward component with respect to q_{goal}, we obtain the reward component with the desired trajectory point of the next time step as the goal:

$$G_2(n) = g_2 \times \tau \times \frac{q_{\text{track}}(n+1) - q_{near}(n)}{\|q_{\text{track}}(n+1) - q_{near}(n)\|} \tag{23}$$

3. Rewarding the generation of new nodes that stay away from obstacles.
   The new node q_{new} is affected by the reward component $G_3(n)$ introduced between the barrier and $q_{near}(n)$ as it is generated, which can be expressed as

$$G_3(n) = \begin{cases} 0, p(n) > p_0 \\ k \times \tau\left(\frac{1}{p(n)} - \frac{1}{p_0}\right)\frac{1}{p(n)^2}\frac{\partial(q_{near} - x_{obstacle})}{\partial q_{near}}, p(n) <= p_0 \end{cases} \tag{24}$$

Summarizing the above analysis, we can visualize the generation process of new node q_{new} by Figure 5.

3.2. Applications of Deep Reinforcement Learning

To counteract the effects of external perturbations and actuator failures on UUV operation, we apply UUV dynamics model training that excludes external perturbations and actuator additive faults when training the network model using deep reinforcement learning. In the actual test environment, we introduced feed-forward compensation to compensate for the estimated perturbation and additive fault terms before the UUV actuators executed the vehicle's action. We rewrite Equation (1) as

$$\begin{cases} \dot{u} = \frac{m_{22}}{m_{11}} vr - \frac{m_{33}}{m_{11}} wq - \frac{1}{m_{11}} f_u - \frac{\lambda_{uh}}{m_{11}} \tau_u \\ \dot{v} = -\frac{m_{11}}{m_{22}} ur - f_v - \frac{1}{m_{22}} \tau_{dv}(t) \\ \dot{w} = \frac{m_{11}}{m_{22}} uq - f_w - \frac{1}{m_{33}} \tau_{dv}(t) \\ \dot{q} = \frac{m_{33} - m_{11}}{m_{55}} uw - f_q - \frac{\rho_e g \nabla G M_L \sin(\theta)}{m_{55}} - \frac{\lambda_{qh}}{m_{55}} \tau_q \\ \dot{r} = \frac{m_{11} - m_{22}}{m_{66}} uv - f_r - \frac{\lambda_{rh}}{m_{66}} \tau_r \end{cases} \tag{25}$$

Equation (25) is used for network model training, in conjunction with the state space in Section 2.2. The state we select for training is:

$$s = [g_s, \tau_{dv}, \tau_{dw}, e_s] \tag{26}$$

where g_s and e_s are consistent with the meanings expressed in Equation (9). Since the UUV does not have actuators in the v and w velocity directions, it cannot offset the effects of perturbations and faults in the form of feed-forward compensation, so we incorporate the perturbations τ_{dv} and τ_{dw} in these two directions into the state space during the training of the network model and train the UUV to output appropriate actions to resist the effects of the perturbations in these two directions.

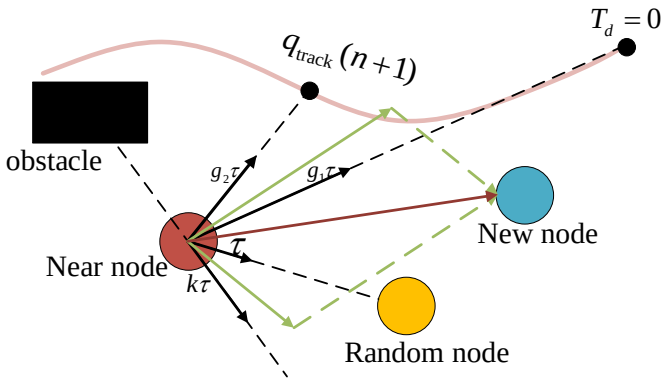


Figure 5. Constructing new nodes by increasing the reward component.

Then, the feed-forward compensation acts directly on the output action of the UUV actuator, which can be represented as

$$a' = \begin{bmatrix} \tau'_u \\ \tau'_q \\ \tau'_r \end{bmatrix} = \begin{bmatrix} \tau_u \\ \tau_q \\ \tau_r \end{bmatrix} - \begin{bmatrix} \hat{\chi}_u \\ \hat{\chi}_q \\ \hat{\chi}_r \end{bmatrix} \tag{27}$$

The compensated force and torque output of the actuator a' ensures that the UUV can safely navigate without disturbances and actuator faults.

Based on the state space of the task environment established in Section 2.2, the architecture of the trajectory tracking task can be represented by Figure 6.

Given the powerful performance of the TD3 algorithm, we embed it in the proposed deep reinforcement learning-based task framework. In the TD3 algorithm, the state-action value function is mainly used to evaluate the current strategy, which we define as

$$Q_{\theta_k}(s_t, a_t) \doteq r(s_t, a_t) + \gamma E_{s_{t+1} \sim p}[V(s_{t+1})],\ k = 1, 2 \tag{28}$$

where $V(s_t)$ is the state value function, and γ is the discount factor. In the state-action value network updating process, a reverse gradient strategy is used, and its objective function can be expressed as follows:

$$y = r(s_t, a_t) + \gamma \left(\min_{k=1,2} Q_{\bar{\theta}_k}(s_{t+1}, a_{t+1} + \varepsilon \middle| \bar{\theta}_k^Q) \right) \tag{29}$$

where ε is a random noise obeying a normal distribution, and the TD3 algorithm introduces two target state-action value networks $Q_{\bar{\theta}_k}(s_t, a_t), k = 1, 2$ and chooses the smaller value of the computation result to be the state-action value of the next state action [30]. Then, its update objective can be expressed as

$$J_Q(\theta_k) = E_{(s_t, a_t) \sim \mho} \left[\frac{1}{2} \left(Q_{\theta_k}(s_t, a_t) - y \right)^2 \right] \tag{30}$$

where $\mho$ is the replay buffer used to store s_{i-1}, a_{i-1}, and s_i, which is shown in Figure 6. Unlike the state-action value function update approach described above, we update the parameters ω of the strategy network $\pi_\omega(a_t|s_t)$ by gradient ascent to make the policy network converge to the vicinity of the maximum value of the Q-values. Its update goal can be expressed as follows:

$$J_\pi(\omega) = E_{s_t \sim \mho} \left[\max_{k=1,2} Q_{\theta_k}(s_t, a_t) \right] \tag{31}$$

In addition, to improve the generalization ability of the policy network, we include the reference value in the training, which can be denoted as

$$l_d = f(\varpi_i, t), i = x_m, y_m, z_m \tag{32}$$

where ϖ_i is a random variable obeying a uniform distribution $U\left(\varepsilon_{\min}^d, \varepsilon_{\max}^d\right)$.

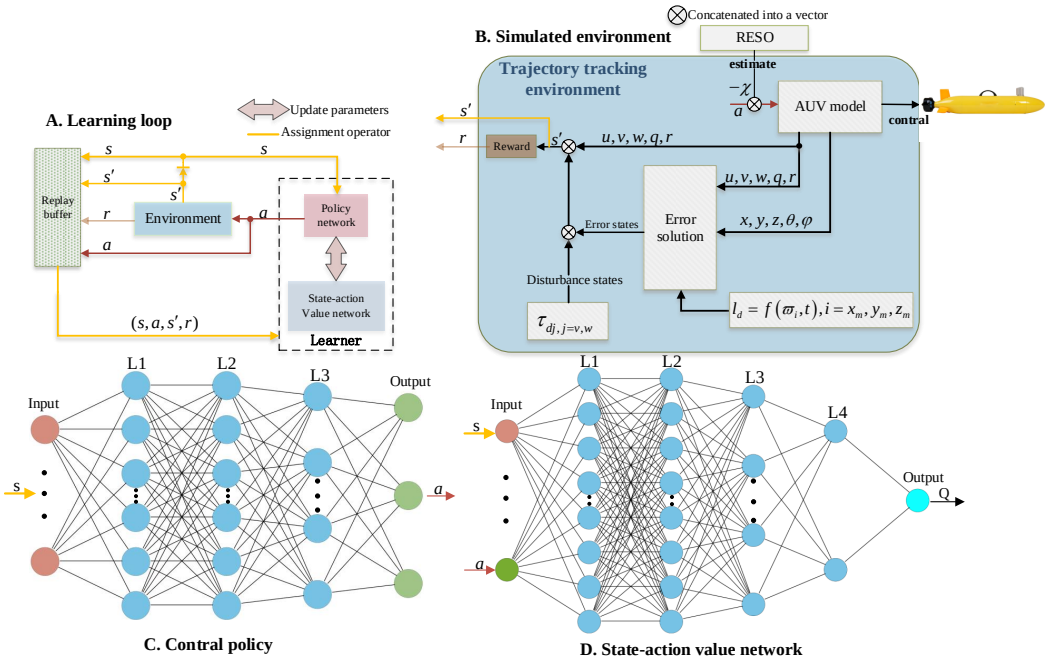


Figure 6. Component representation of trajectory tracking task architecture. (**A**) Interaction between environment and network model. (**B**) The trajectory tracking task simulation environment, whose composition is consistent with that presented in Section 2.2. (**C**) The control policy and its target network are the mapping from the state space to the action space. (**D**) Calculation of Q-values using state-action value networks and their target networks.

At this point, the task strategies of trajectory tracking and local planning have been established. Next, incorporating the description of the event triggering mechanism in Section 2.3, we can obtain the execution rules of these two task strategies in the continuous task space, which are represented by the following pseudo-code (Algorithm 1).

Algorithm 1 Pseudo-code of TD3 for UUV path tracking with local path planning

Initialize the interactive environment
Randomly initialize state-action value networks Q_{θ_1} and Q_{θ_2} and policy network π_ω with parameter vectors $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$, and $\boldsymbol{\omega}$
Initialize target state-action value networks $Q_{\bar{\theta}_1}$ and $Q_{\bar{\theta}_2}$ with parameter vectors $\bar{\boldsymbol{\theta}}_1 \leftarrow \boldsymbol{\theta}_1$ and $\bar{\boldsymbol{\theta}}_2 \leftarrow \boldsymbol{\theta}_2$
Initialize replay buffer $\mathbb{D}_1$ to capacity $\boldsymbol{B}$ and empty it
Initialize global shared step counter $C \leftarrow 0$ and maximum time steps of each episode T
Set the maximum number of iterations C_{max} and some other necessary parameters (refer to Section 4.1)
for $C \in \{0, 1, 2, \cdots, C_{max}\}$ **do**
    Reset step counter for each episode $t \leftarrow 0$
    Reset the environment, $E_d \leftarrow 0$, $s_t \leftarrow s_0$, and so on
    repeat
        Perform action with exploration noise $a \leftarrow \pi_\omega(s) + \chi,\ \chi \sim \mathcal{N}(0, \sigma)$
        Obtain partial states $s_{(t+1)}$ and partial reward r
        Store samples in the replay buffer $\mathbb{D} \leftarrow \mathbb{D} \cup \left\{ \left(\boldsymbol{s_t}, \boldsymbol{a}, \boldsymbol{s_{(t+1)}}, r \right) \right\}$
        if $|\mathbb{D}| == \boldsymbol{B}$ **then**
            Sample b_s samples $\left(\boldsymbol{s_t}, \boldsymbol{a}, \boldsymbol{s_{(t+1)}}, r \right)$ from $\mathbb{D}$
            Update $\boldsymbol{\theta}_k \leftarrow \boldsymbol{\theta}_k - \lambda_{\boldsymbol{\theta}} \hat{\nabla}_{\boldsymbol{\theta}_k} J_Q(\boldsymbol{\theta}_k)$, for $k = 1, 2$
            Update $\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} - \lambda_{\boldsymbol{\omega}} \hat{\nabla}_{\boldsymbol{\omega}} J_\pi(\boldsymbol{\omega})$
            Update $\bar{\boldsymbol{\theta}}_k \leftarrow \varrho \boldsymbol{\theta}_k + (1 - \varrho) \bar{\boldsymbol{\theta}}_k$, for $k = 1, 2$
        end if
        $\boldsymbol{s_t} \leftarrow \boldsymbol{s_{(t+1)}}$
        $t \leftarrow t + 1$
    until $t == T$
end for
Start continuous task testing
Initialize environment, including desired trajectory, obstacle states, and RRT R
repeat
    Execute the action a output from the trained policy network π_ω, tracking the desired trajectory
    if $T_d == 1$ **then**
        for $i \in \{1, 2, \cdots, n\}$ **do**
            $q_{rand} \leftarrow Sample(i)$
            $q_{near} \leftarrow Nearest(tree, q_{rand})$
            $(q_{new}, U_{new}) \leftarrow Steer(q_{near}, q_{rand})$ refer to Section 3.1
            Judge the relationship between $\bar{\partial}(n)$ and $\bar{\partial}_{\max}$, $\bar{\beta}(n)$ and $\bar{\beta}_{\max}$
            Update q_{new} by Equation (20)
            if $Obstaclefree(q_{new})$ **then** $R \leftarrow q_{new}$
            end if
            $T_d == 0$
        end for
    end if
until
Continue to complete the trajectory tracking task

Remark 2. *Please refer to [30] for a detailed description of the TD3 algorithm, which is not developed in this paper. The relevant parameters of the algorithm during the simulation will be given in Section 4.1.*

4. Simulation

In this section, the UUV's trajectory tracking and local planning capabilities and continuous task completion capabilities will be tested in the framework of the above tasks.

4.1. Trajectory Tracking Capability

The parameter settings for the deep reinforcement learning algorithm were consistent before the simulation began. As shown in Table 1, for the 3D trajectory tracking task, we set the capacity of replay memory B to 100,000 and the batch size b_s to 128. The learning rate was then set to 0.0003. Next, we selected the desired trajectory for model training as follows:

$$\begin{cases} x_d = 8\sin((0.10 + 0.1\varpi_{x_m})t) \\ y_d = 10\cos\big((0.10 + 0.15\varpi_{y_m})t\big) \\ z_d = -(0.15 + 0.1\varpi_{z_m})t - 3 \end{cases} \tag{33}$$

where ϖ_{x_m}, ϖ_{y_m}, and ϖ_{z_m} are random variables with the same meaning as ϖ_i in Equation (32). At the same time, the perturbations in both directions v and w required for the state space during training were set as follows:

$$\tau_{dj} = \vartheta_j(1 - \gamma_j)\cos\big((\omega_j + \xi_j)t + \kappa_j\big)j = u, v, w, q, r \tag{34}$$

Table 1. The values of the shared parameters.

Parameter	Value and Description
m_{11}, m_{22}, m_{33}, m_{55}, m_{66}	25.0, 17.5, 30.0, 22.5, 15.0
$\rho g \nabla GM_L$	9.7
λ_{uh}	0.8
λ_{qh}	0.8
λ_{rh}	0.8
$\bar{\omega}_{uh}$	$0.5\sin(0.1t)$
$\bar{\omega}_{qh}$	$0.4\cos(0.1t)$
$\bar{\omega}_{rh}$	$0.3\sin(0.1t)$
Policy network (full connection layer)	$\rightarrow$ 128 $\rightarrow$ 64 $\rightarrow$ 32 $\rightarrow$
State-action value network	$\rightarrow$ 320 $\rightarrow$ 320 $\rightarrow$ 160 $\rightarrow$ 80 $\rightarrow$
$\varepsilon^d_{\min}$, $\varepsilon^d_{\max}$	0.0, 1.0
$\vartheta_{j=u,v,w,q,r}$	3.0, 1.5, 2.5, 1.5, 2.0
$\gamma_{j=u,v,w,q,r}$	0.0
$\omega_{j=u,v,w,q,r}$	0.9, 0.5, 0.4, 0.5, 0.6
$\xi_{j=u,v,w,q,r}$	0.0
$\kappa_{j=u,v,w,q,r}$	1.0
$k_{1j=u,v,w,q,r}$	2.4, 4.2, 1.4, 2.2, 3.1
$k_{2j=u,v,w,q,r}$	1.1, 0.8, 0.5, 1.3, 0.8

Table 1. *Cont.*

Parameter	Value and Description
$\partial_{j=u,v,w,q,r}$	0.5, 0.5, 0.5, 0.5, 0.4
ζ_s, ζ_p	0.01, 0.001
γ	0.99
σ	1.0
$\boldsymbol{B}$	100,000
b_s	128

For training, we only needed the perturbations in the direction of the two velocities u and v as input states, whose parameters were set as $\gamma_v, \gamma_w, \xi_v, \xi_w, \kappa_v, \kappa_w \sim U\left(\varepsilon^d_{\min}, \varepsilon^d_{\max}\right)$, with the values updated each training episode. Meanwhile, we set $\vartheta_v = 1.5, \vartheta_w = 1.6, \omega_v = 0.8, \omega_w = 0.5$. In the actual test, we set the error state based on Equation (8): $e_s = [\rho_d - \rho, \beta_d - \beta, \alpha_d - \alpha]^T$, where ρ_d is 1.0, β_d and α_d are both 0, and the desired trajectory is set to

$$\begin{cases} x_d = 10\sin(0.1t) \\ y_d = 10\cos(0.10t) \\ z_d = -0.1t - 5 \end{cases} \tag{35}$$

In addition, the parameters related to the amount of disturbance in the actual test could all be set to constant values; please refer to the data in Table 1.

The training process is illustrated by Figure 7, where the average reward was calculated by evaluating the learned action strategy every 100 time steps, such that in each episode we evaluated the action strategy 10 times and finally calculated the average reward for each episode. In Figure 7, the top and bottom of the shaded area indicate the maximum and minimum of the average reward over the 10 assessments, respectively. The end of the 100th episode was taken as the dividing line, before which the process of data collection for the stochastic control strategy was carried out, and after which the action strategy started to be optimized iteratively by gradient descent until the optimal control strategy was obtained. The entire training process took 8.3 h on a computer with a CPU (Inter(R) Core(TM) i5-12400h @2.5GHz made in Portland, OR, United States) and a GPU (NVIDIA GeForce GTX 3070ti 8GB made in Santa Clara, CA, United States). Comparing the deep reinforcement learning algorithms, it is easy to see that the TD3 algorithm rewarded faster convergence and better training stability than the DDPG algorithm (the DDPG algorithm showed individually larger fluctuations near the end of training).

The results of the test are shown in Figures 8–11. For comparison with the classical control methods mentioned in Section 1, we reproduced the trajectory tracking controller based on the LOS method in [3]. Additionally, in the context of deep reinforcement learning, in order to verify the convergence and stability performance of the TD3 algorithm in this task, the DDPG algorithm was used as the benchmark algorithm in the same task environment for comparison. In Figures 8 and 9, it is clear that the TD3 algorithm approached the desired trajectory faster and demonstrated better convergence performance and less fluctuation relative to the classical and DDPG algorithms. The problem with the classical algorithms was that they showed poor convergence performance and took a long time to track the target, while the DDPG algorithm, although equal to the TD3 algorithm in terms of tracking speed, had poor stability. In addition, we show the compensated action output values in Figure 10. The DDPG algorithm presented a large-amplitude action output, which would cause dysfunction in the UUV propulsion structure. The estimations of the designed reduced-order extended state observer are shown in Figure 11: the first graph shows the velocity estimates for the UUV with five degrees of freedom, subsequent

graphs show the estimates of the combined perturbation and additive fault terms under the five degrees of freedom condition, where the red dashed lines all indicate actual values. It is easy to see that the observer quickly converged to the target value after a short period of oscillation.

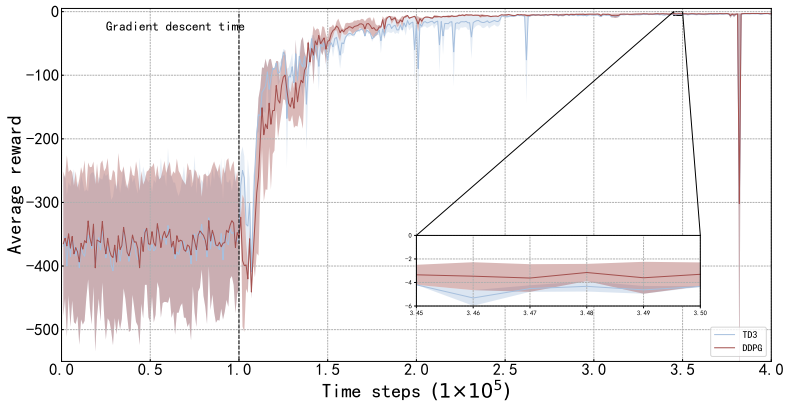


Figure 7. Average reward in 3D trajectory tracking training.

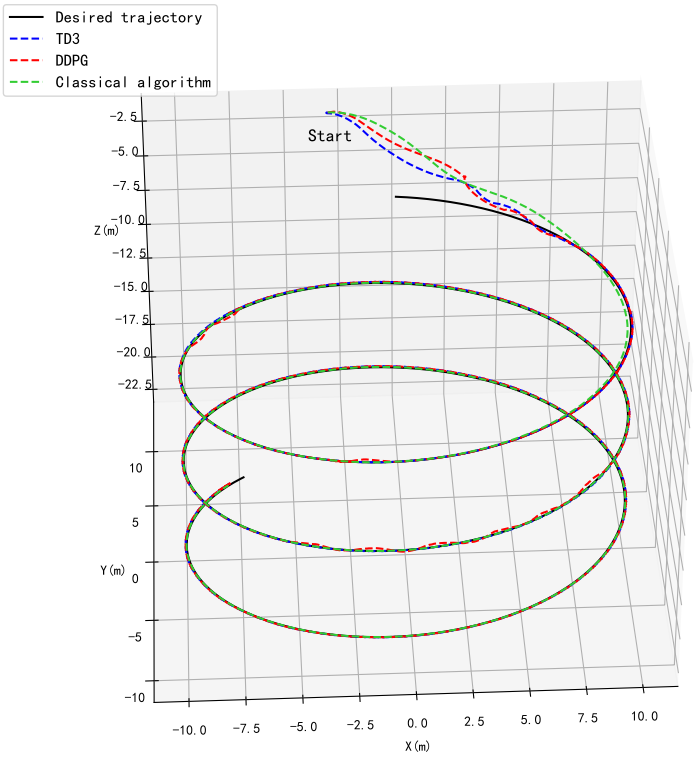


Figure 8. Desired and actual trajectories.

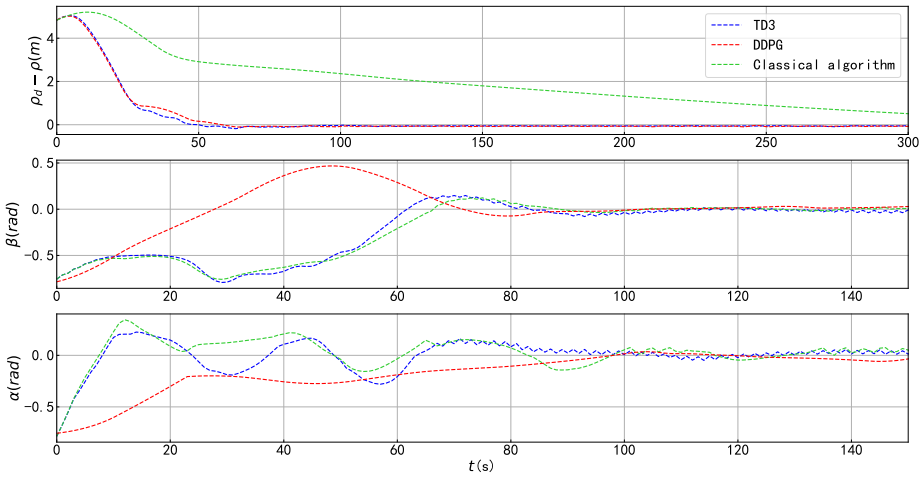


Figure 9. Trajectory tracking errors.

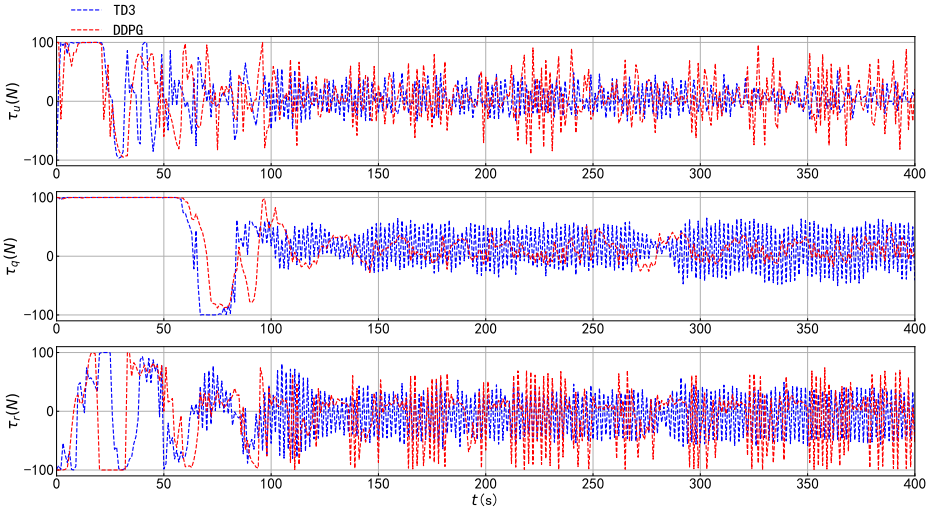


Figure 10. Output forces and moments.

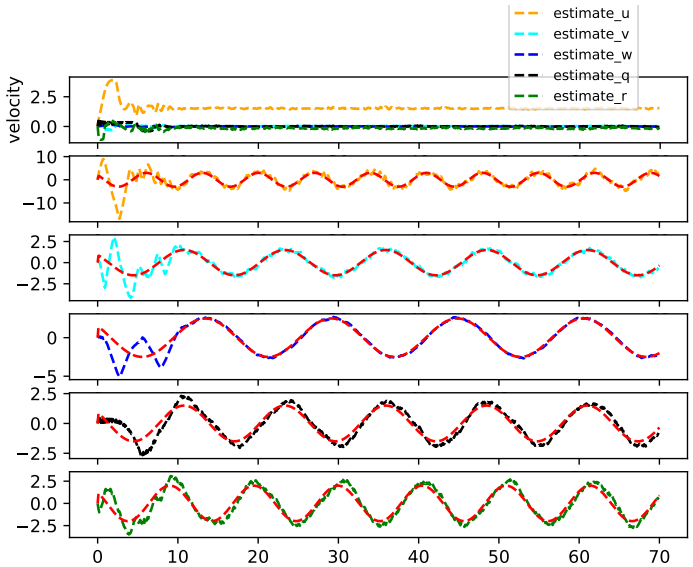
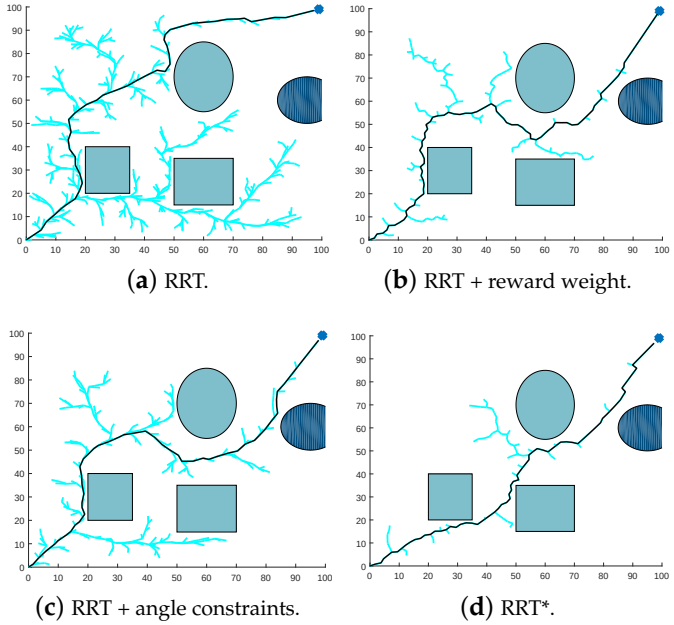


Figure 11. Estimated vs. true values.

4.2. Localized Path Planning Capability

To test the local path planning ability of the UUV, we designed a multi-obstacle environment and compared the pre- and post-improvement RRT algorithms. We assumed that the simulation environment was a space of 100 m × 100 m; the detection radius of the UUV was 10 m; and the starting and ending coordinates were (0, 0) and (99, 99), respectively. The results are shown in Figure 12, where RRT* denotes the improved RRT algorithm obtained in Section 3.1. It is easy to see that the convergence of the spanning tree toward the goal point was greatly improved after the introduction of the gravitational component, and the path smoothness was guaranteed after the addition of the angle restriction. Figure 13 demonstrates the comparison of the above algorithms in terms of search speed and exploration degree. The exploration efficiency and success rate for finding the shortest collision-free path of the RRT* algorithm proposed in this paper were greatly improved in the multi-obstacle environment.



(**a**) RRT. (**b**) RRT + reward weight.

(**c**) RRT + angle constraints. (**d**) RRT*.

Figure 12. Localized path planning in multi-obstacle environment.

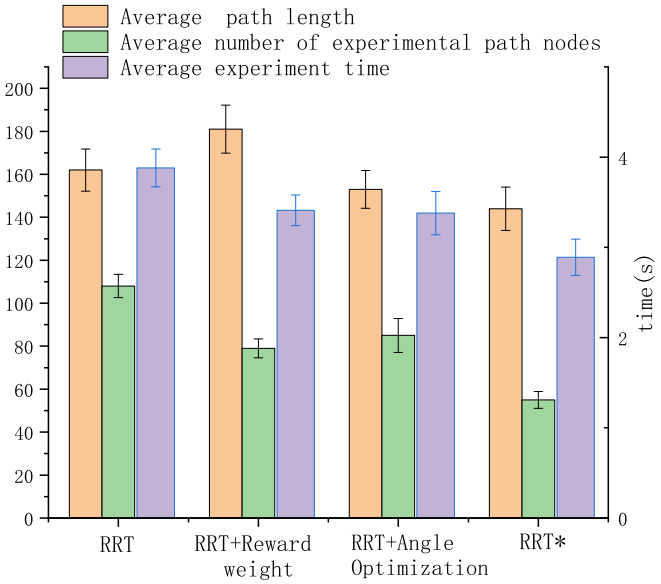


Figure 13. Performance comparison between basic RRT and the improved algorithm.

4.3. Trajectory Tracking Combined with Localized Path Planning Capabilities

Based on the event triggering mechanism in Section 2.3, we completed a joint test of the UUV trajectory tracking and local planning capabilities in a 3D environment. We set the desired trajectories as follows:

$$\begin{cases} x_d = \begin{cases} 2.0 + 0.7t, 0 \leq \mathrm{t} < 20 \\ -7.0 + 1.3t, 20 \leq \mathrm{t} < 60 \end{cases} \\ y_d = \begin{cases} 9.0 + 1.3t, 0 \leq \mathrm{t} < 20 \\ -6.7 + 2.0t, 20 \leq \mathrm{t} < 60 \end{cases} \\ z_d = \begin{cases} -3.0 - 0.4t, 0 \leq \mathrm{t} < 20 \\ 6.3 - 0.9t, 20 \leq \mathrm{t} < 60 \end{cases} \end{cases} \tag{36}$$

For the desired trajectory described in Equation (36), the tracking results of the three algorithms (TD3, DDPG, and the classical algorithm) are represented in Figures 14–16 and were basically the same as the results in Section 4.1. However, the classical algorithm deviated at about the 210th time step and moved away from the desired trajectory. The TD3 algorithm outperformed the DDPG algorithm in terms of tracking speed, tracking stability, and the magnitude of the output force and moment. In order to fully test the performance of the UUV in the joint task of trajectory tracking and local planning, we established continuous obstacle avoidance and complex obstacle environments, respectively, on the basis of the above tracking trajectories, and the experimental results are analyzed below.

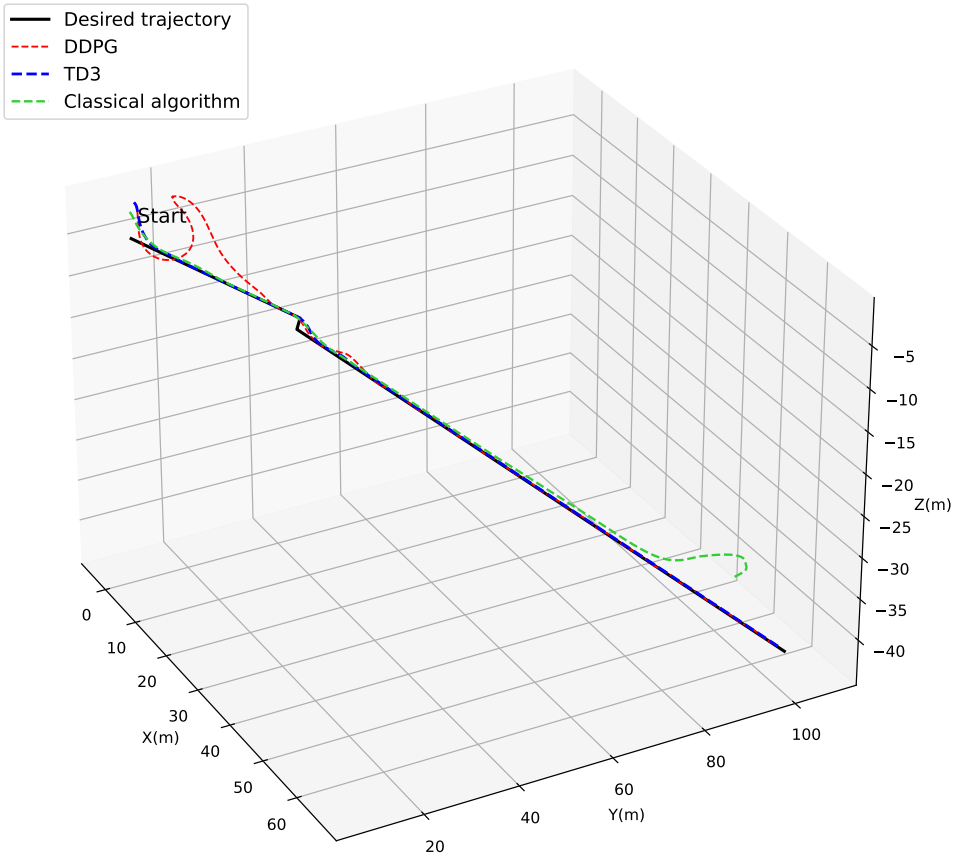


Figure 14. Tracking mission trajectories.

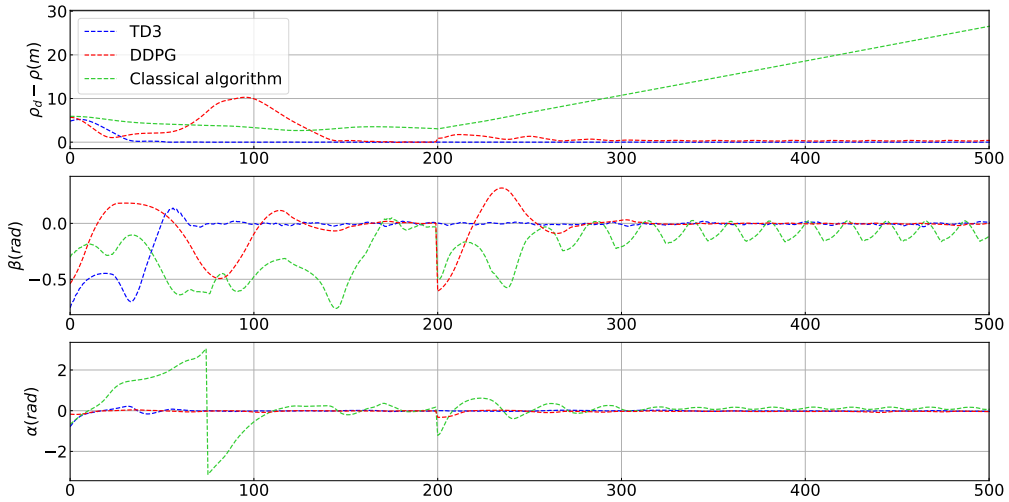


Figure 15. Trajectory tracking errors.

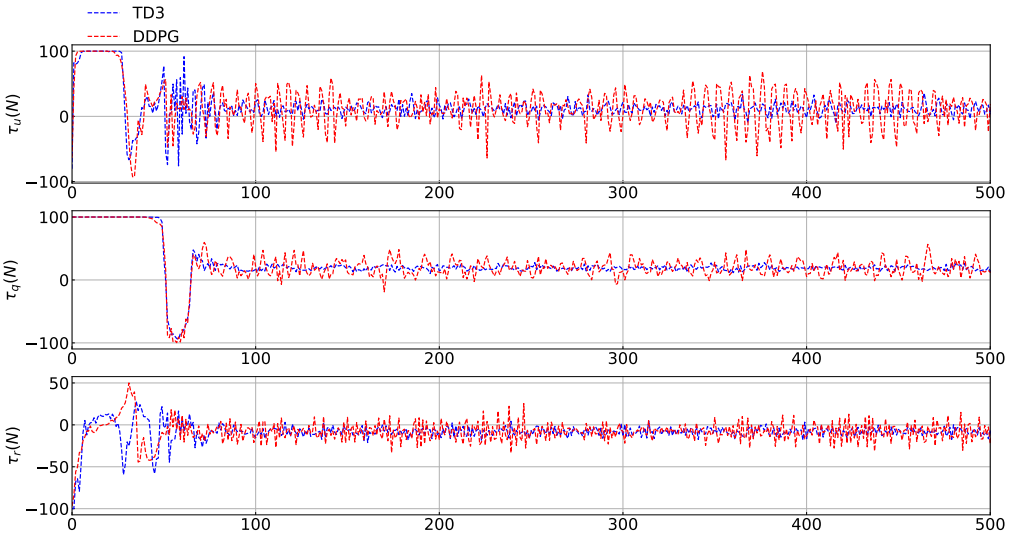


Figure 16. Output forces and moments.

As shown in Figures 17 and 18, in the continuous obstacle environment, the four RRT-based algorithms realized the switching of task states and completed the planning of local collision-free paths through the event triggering mechanism. Due to some randomness in the RRT algorithm, when the reward component was not added, the local path was generated away from the desired direction at a distance larger than half of the sphere. However, after the addition of the reward component, the local path was generated closer to the desired trajectory and the target point. Meanwhile, the smoothness of the planned paths was improved after angle optimization was implemented. Figures 19 and 20 show the joint experiment in a complex-obstacle environment, where the local planning task was turned on after the UUV detected an obstacle ($T_d == 1$). The four algorithms under the RRT framework planned different collision-free paths before T_d became 0. It is easy to see in the side view of Figure 19 that the paths planned by the basic RRT and RRT with angular constraints algorithms were relatively long and did not find narrow passages between obstacles. Conversely, after adding the reward component, as shown in the top view of Figure 20, the generation of the localized path moved towards the desired trajectory and successfully searched for the narrow passage. The effectiveness of the proposed algorithm was confirmed.

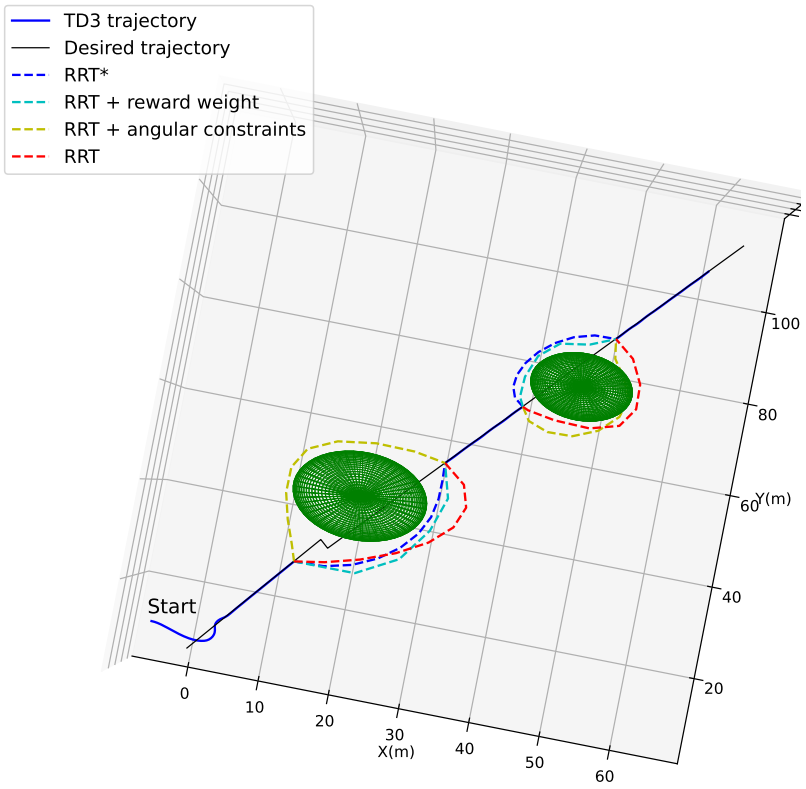


Figure 17. Joint testing in a continuous obstacle environment (top view).

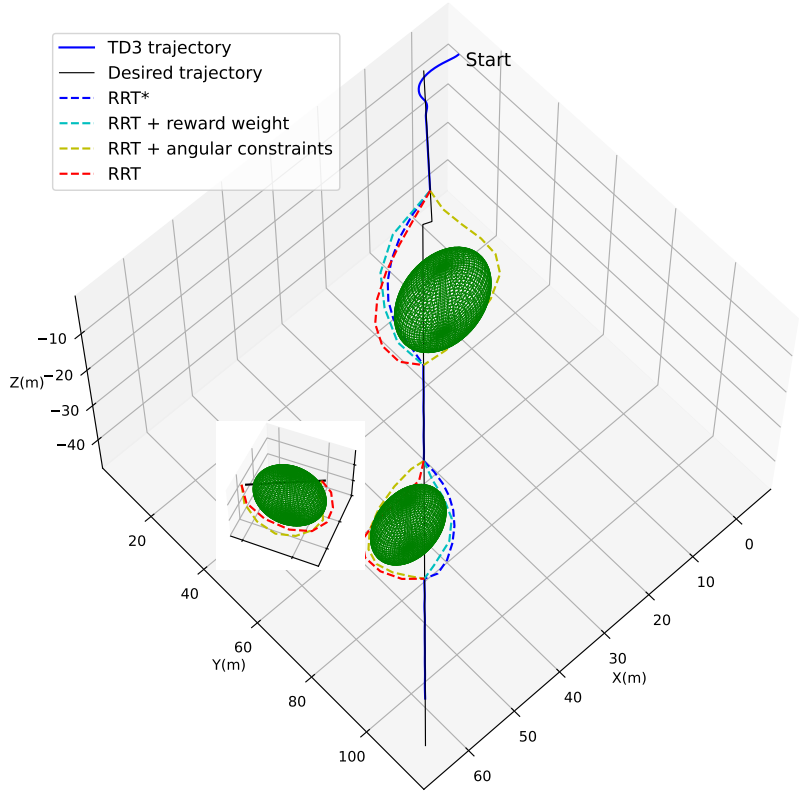


Figure 18. Joint testing in a continuous obstacle environment (side view).

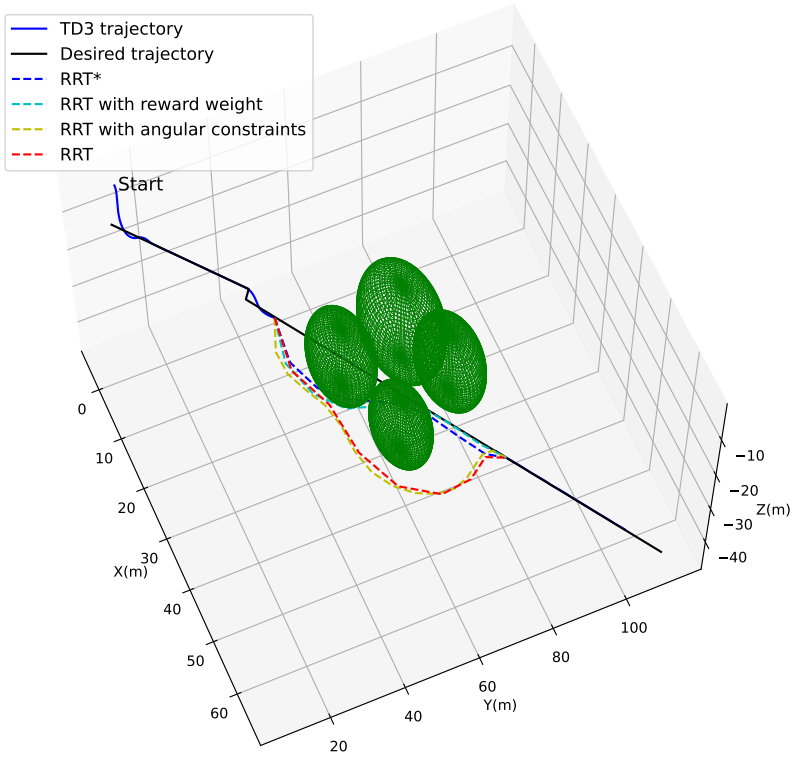


Figure 19. Joint testing in multi-obstacle environment (side view).

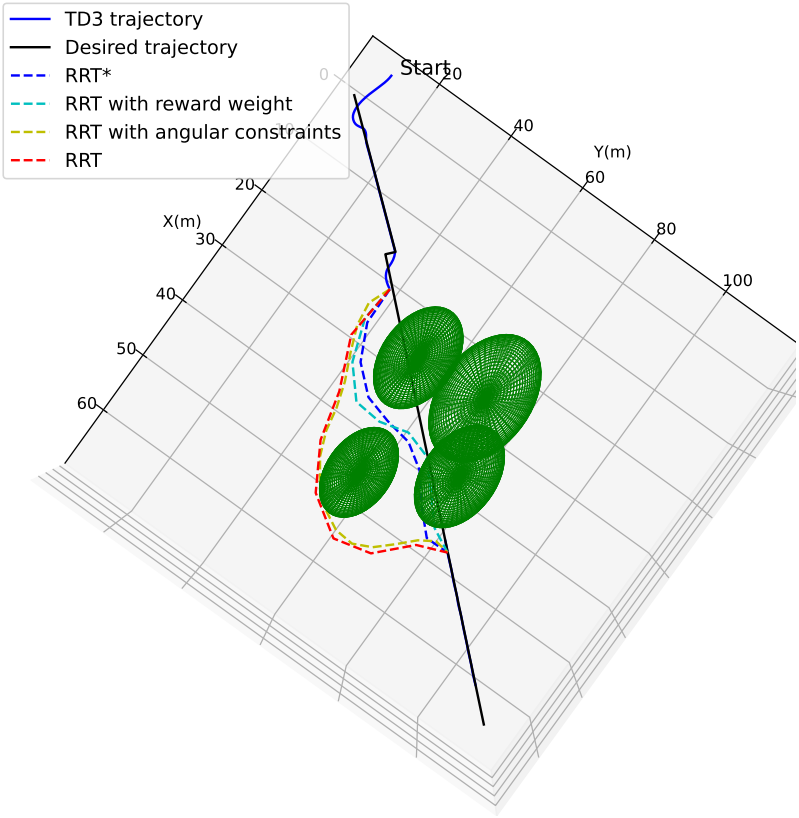


Figure 20. Joint testing in multi-obstacle environment (top view).

5. Conclusions

In this paper, a framework for the joint task of UUV underwater trajectory tracking and local path planning was proposed. In this framework, the event triggering mechanism was

utilized to realize the switching of task states, and a trajectory tracking controller under a deep reinforcement learning framework was combined with an RRT algorithm to realize local planning in multi-obstacle environments. We introduced two kinds of reward components and angular constraints into the RRT algorithm, which directed the spanning tree towards the target point and deferred the application of the desired trajectory, respectively, preventing the UUV from encountering steering difficulties in the real environment. In order to resist the effects of external perturbations, we proposed a way to observe the perturbations using a reduced-order extended observer and incorporate reinforcement learning features for targeted compensation. Finally, simulation experiments on the UUV's trajectory tracking ability, local planning ability, and continuous task completion ability were conducted under our proposed framework, which was compared with the classical control algorithm and DDPG algorithm described in Section 1. The results showed that our method achieved better convergence to the desired trajectories and better search performance during planning.

Although this paper demonstrates the feasibility and superiority of our methodology in the theoretical and simulation domains, and the treatment of actuator faults and external perturbations gives the methodology the potential to be applied in a practical environment, the engineering application domain remains the focus of our next work. In addition, the incorporation of the RRT method is not efficient for underwater exploration. In the future, we will use deep reinforcement learning as a framework to explore more efficient and safer methods to tackle real problems in engineering.

Author Contributions: methodology, Z.W.; software, X.Z.; validation, H.D. and H.C.; formal analysis, X.Z.; investigation, Z.W.; writing—original draft preparation, Z.W., X.Z., and H.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Natural Science Foundation of Shandong Province (grant number ZR2022ME104).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

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

Appendix A

Convergence proofs for reduced-order extended observer.

Proof. Let $\delta = \hat{v} - v$, $\delta_1 = \hat{\chi} - v_0$, $\dot{v}_0 = V(t)$. According to Equations (16) and (17), we obtain

$$\begin{cases} \dot{\delta} = \delta_1 - \beta_1 \delta \\ \dot{\delta}_1 = -V(t) - \beta_2 |\delta|^{\partial} \mathrm{sgn}(\delta) \end{cases} \tag{A1}$$

where $V(t)$ is bounded and satisfies $|V(t)| < h_0$. Then, (δ, δ_1) can be divided into five parts, as follows:

$$\begin{cases} V_0 = \left\{ (\delta, \delta_1) \middle| |\delta| < \Gamma_0, \frac{\varepsilon}{2}(\delta - \Gamma_0) < \delta_1 \leq \frac{\varepsilon}{2}(\delta + \Gamma_0) \right\} \\ V_1 = \left\{ (\delta, \delta_1) \middle| \delta > \Gamma_0, 0 \leq \delta_1 \leq \varepsilon\delta \right\} \\ V_2 = \left\{ (\delta, \delta_1) \middle| \delta_1 > 0, \delta_1 \geq \frac{\varepsilon}{2}(\delta + \Gamma_0), \delta_1 \geq \varepsilon\delta \right\} \\ V_3 = \left\{ (\delta, \delta_1) \middle| \delta < -\Gamma_0, 0 \geq \delta_1 \geq \varepsilon\delta \right\} \\ V_4 = \left\{ (\delta, \delta_1) \middle| \delta_1 < 0, \delta_1 \leq \frac{\varepsilon}{2}(\delta - \Gamma_0), \delta_1 \leq \varepsilon\delta \right\} \end{cases} \tag{A2}$$

Next, the following smooth Lyapunov functions are constructed for each of the above five discontinuous curves:

$$G = \begin{cases} 0, (\delta, \delta_1) \in V_0 \\ \frac{\varepsilon}{2}(\delta - \Gamma_0), (\delta, \delta_1) \in V_1 \\ \delta_1 - \frac{\varepsilon}{2}(\delta + \Gamma_0), (\delta, \delta_1) \in V_2 \\ -\frac{\varepsilon}{2}(\delta + \Gamma_0), (\delta, \delta_1) \in V_3 \\ -\delta_1 + \frac{\varepsilon}{2}(\delta - \Gamma_0), (\delta, \delta_1) \in V_3 \end{cases} \tag{A3}$$

The Lyapunov function G is positive definite outside V_0. In the region V_1, because $\delta > \Gamma_0$ and $0 \le \delta_1 \le \varepsilon\delta$, we have

$$\frac{dG}{dt} = \frac{\varepsilon}{2}(\delta_1 - \beta_1\delta) = \frac{\varepsilon}{2}[\delta_1 - \varepsilon\delta + (\varepsilon - \beta_1)\delta] \le \frac{\varepsilon}{2}(\varepsilon - \beta_1)\delta \tag{A4}$$

Take $\varepsilon = \beta_1 - \kappa$. If κ is sufficiently small, we can easily obtain that

$$\frac{dG}{dt} \le -\kappa\frac{\beta_1 - \kappa}{2}\delta < 0 \tag{A5}$$

In the same way, it can be proved that $\frac{dG}{dt} < 0$ in the region V_3. Under the conditions $\varepsilon = \beta_1 - \kappa$, $\Gamma_0 < \Gamma$ and $|\delta| \le \Gamma$, in order to make the formula

$$\frac{dG}{dt} = -\frac{\varepsilon}{2}(\delta_1 - \beta_1\delta) - V(t) - \beta_2|\delta|^\partial \mathrm{sgn}(\delta) \le \frac{\varepsilon}{2}\left(\beta_1 - \frac{\varepsilon}{2}\right)\delta - \frac{\varepsilon^2}{4}\Gamma + h_0 - \beta_2|\delta|^\alpha \mathrm{sgn}(\delta) \tag{A6}$$

negative, the following inequality needs to be satisfied:

$$\frac{\varepsilon}{2}\left(\beta_1 - \frac{\varepsilon}{2}\right)\delta < \beta_2|\delta|^\alpha \mathrm{sgn}(\delta) - h_0 \tag{A7}$$

Substituting $\varepsilon = \beta_1 - \kappa$ into Equation (A7), we obtain

$$\frac{\beta_1^2 - \kappa^2}{4}\left[\delta - \frac{(\beta_1 - \kappa)^2}{\beta_1^2 - \kappa^2}\Gamma\right] < \beta_2\left[|\delta|^\alpha \mathrm{sgn}(\delta) - \frac{h_0}{\beta_2}\right] \tag{A8}$$

If κ is sufficiently small, Equation (A8) is rewritten as

$$\frac{\beta_1^2}{4}[\delta - \Gamma] < \beta_2\left[|\delta|^\alpha \mathrm{sgn}(\delta) - \frac{h_0}{\beta_2}\right] \tag{A9}$$

After analysis, the following two inequalities can be obtained:

$$\begin{cases} \Gamma > \left(\frac{h_0}{\beta_2}\right)^{\frac{1}{\alpha}} \\ \Gamma > \frac{4h_0}{{\beta_1}^2} \end{cases} \tag{A10}$$

Inequality (A9) holds when the above inequalities are satisfied. According to the above analysis, as long as Γ satisfies Equation (A10) and κ is sufficiently small, Equation (A9) holds in the range $|\delta| \le \Gamma$. Based on the above detailed analysis, when Equation (A10) is satisfied, $\frac{dG}{dt}$ holds in the region V_2. In the same way, it can be proved that $\frac{dG}{dt} < 0$ in the region V_4. This completes the proof. □

References

1. Zhang, C.; Cheng, P.; Du, B.; Dong, B.; Zhang, W. AUV path tracking with real-time obstacle avoidance via reinforcement learning under adaptive constraints. *Ocean Eng.* **2022**, *256*, 111453. [CrossRef]
2. Lin, X.; Tian, W.; Zhang, W.; Li, Z.; Zhang, C. The fault-tolerant consensus strategy for leaderless Multi-AUV system on heterogeneous condensation topology. *Ocean Eng.* **2022**, *245*, 110541. [CrossRef]
3. Shojaei, K.; Dolatshahi, M. Line-of-sight target tracking control of underactuated autonomous underwater vehicles. *Ocean Eng.* **2017**, *133*, 244–252. [CrossRef]
4. Shen, C.; Shi, Y.; Buckham, B. Trajectory Tracking Control of an Autonomous Underwater Vehicle Using Lyapunov-Based Model Predictive Control. *IEEE Trans. Ind. Electron.* **2018**, *65*, 5796–5805. [CrossRef]
5. Lamraoui, H.C.; Qidan, Z. Path following control of fully-actuated autonomous underwater vehicle in presence of fast-varying disturbances. *Appl. Ocean Res.* **2019**, *86*, 40–46. [CrossRef]

6. Vu, M.T.; Le Thanh, H.N.N.; Huynh, T.T.; Thang, Q.; Duc, T.; Hoang, Q.D.; Le, T.H. Station-keeping control of a hovering over-actuated autonomous underwater vehicle under ocean current effects and model uncertainties in horizontal plane. *IEEE Access* **2021**, *9*, 6855–6867. [CrossRef]
7. Le, T.H.; Thanh, H.L.N.N.; Huynh, T.T.; Van, M.; Hoang, Q.D.; Do, T.D.; Vu, M.T. Robust position control of an over-actuated underwater vehicle under model uncertainties and ocean current effects using dynamic sliding mode surface and optimal allocation control. *Sensors* **2021**, *21*, 747.
8. Yu, C.; Xiang, X.; Wilson, P.A.; Zhang, Q. Guidance-Error-Based Robust Fuzzy Adaptive Control for Bottom Following of a Flight-Style AUV With Saturated Actuator Dynamics. *IEEE Trans. Cybern.* **2020**, *50*, 1887–1899. [CrossRef]
9. Carlucho, I.; De Paula, M.; Wang, S.; Menna, B.V.; Petillot, Y.R.; Acosta, G.G. AUV position tracking control using end-to-end deep reinforcement learning. In Proceedings of the OCEANS 2018 MTS/IEEE Charleston, Charleston, SC, USA, 22–25 October 2018; IEEE: New York, NY, USA, 2018; pp. 1–8.
10. Mao, Y.; Gao, F.; Zhang, Q.; Yang, Z. An AUV target-tracking method combining imitation learning and deep reinforcement learning. *J. Mar. Sci. Eng.* **2022**, *10*, 383. [CrossRef]
11. Carlucho, I.; De Paula, M.; Wang, S.; Petillot, Y.; Acosta, G.G. Adaptive low-level control of autonomous underwater vehicles using deep reinforcement learning. *Robot. Auton. Syst.* **2018**, *107*, 71–86. [CrossRef]
12. Jiang, J.; Zhang, R.; Fang, Y.; Wang, X. Research on motion attitude control of under-actuated autonomous underwater vehicle based on deep reinforcement learning. *J. Phys. Conf. Ser.* **2020**, *1693*, 012206. [CrossRef]
13. Fang, Y.; Huang, Z.; Pu, J.; Zhang, J. AUV position tracking and trajectory control based on fast-deployed deep reinforcement learning method. *Ocean Eng.* **2022**, *245*, 110452. [CrossRef]
14. Liu, Y.; Wang, F.; Lv, Z.; Cao, K.; Lin, Y. Pixel-to-Action Policy for Underwater Pipeline Following via Deep Reinforcement Learning. In Proceedings of the 2018 IEEE International Conference of Intelligent Robotic and Control Engineering (IRCE), Lanzhou, China, 24–27 August 2018; pp. 135–139. [CrossRef]
15. Yan, Z.; Yan, J.; Wu, Y.; Cai, S.; Wang, H. A novel reinforcement learning based tuna swarm optimization algorithm for autonomous underwater vehicle path planning. *Math. Comput. Simul.* **2023**, *209*, 55–86. [CrossRef]
16. Zheng, Y.J. Water wave optimization: A new nature-inspired metaheuristic. *Comput. Oper. Res.* **2015**, *55*, 1–11. [CrossRef]
17. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
18. Neshat, M.; Sepidnam, G.; Sargolzaei, M.; Toosi, A.N. Artificial fish swarm algorithm: A survey of the state-of-the-art, hybridization, combinatorial and indicative applications. *Artif. Intell. Rev.* **2014**, *42*, 965–997. [CrossRef]
19. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
20. Li, J.; Li, C.; Chen, T.; Zhang, Y. Improved RRT algorithm for AUV target search in unknown 3D environment.*J. Mar. Sci. Eng.* **2022**, *10*, 826. [CrossRef]
21. Hong, L.; Song, C.; Yang, P.; Cui, W. Two-Layer Path Planner for AUVs Based on the Improved AAF-RRT Algorithm. *J. Mar. Sci. Appl.* **2022**, *21*, 102–115. [CrossRef]
22. Yu, L.; Wei, Z.; Wang, Z.; Hu, Y.; Wang, H. Path optimization of AUV based on smooth-RRT algorithm. In Proceedings of the 2017 IEEE International Conference on Mechatronics and Automation (ICMA), Kagawa, Japan, 6–9 August 2017; IEEE: New York, NY, USA, 2017; pp. 1498–1502.
23. Huang, F.; Xu, J.; Wu, D.; Cui, Y.; Yan, Z.; Xing, W.; Zhang, X. A general motion controller based on deep reinforcement learning for an autonomous underwater vehicle with unknown disturbances. *Eng. Appl. Artif. Intell.* **2023**, *117*, 105589. [CrossRef]
24. Fossen, T.I. *Handbook of Marine Craft Hydrodynamics and Motion Control*; John Wiley & Sons: Hoboken, NJ, USA, 2011.
25. Xu, J.; Cui, Y.; Yan, Z.; Huang, F.; Du, X.; Wu, D. Event-triggered adaptive target tracking control for an underactuated autonomous underwater vehicle with actuator faults. *J. Frankl. Inst.* **2023**, *360*, 2867–2892. [CrossRef]
26. Tian, G. Reduced-Order Extended State Observer and Frequency Response Analysis. Master's Thesis, Cleveland State University, Cleveland, OH, USA, 2007.
27. Li, J.; Zhai, X.; Xu, J.; Li, C. Target search algorithm for AUV based on real-time perception maps in unknown environment. *Machines* **2021**, *9*, 147. [CrossRef]
28. Li, J.; Yang, C. AUV path planning based on improved RRT and Bezier curve optimization. In Proceedings of the 2020 IEEE International Conference on Mechatronics and Automation (ICMA), Beijing, China, 2–5 August 2020; IEEE: New York, NY, USA, 2020; pp. 1359–1364.
29. Huang, S. Path planning based on mixed algorithm of RRT and artificial potential field method. In Proceedings of the 2021 4th International Conference on Intelligent Robotics and Control Engineering (IRCE), Lanzhou, China, 18–20 September 2021; IEEE: New York, NY, USA, 2021; pp. 149–155.
30. Fujimoto, S.; Hoof, H.; Meger, D. Addressing function approximation error in actor-critic methods. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; PMLR: Baltimore, MA, USA, 2018; pp. 1587–1596.

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