drones

MDPI

Article

Model-Based Offline Reinforcement Learning for AUV Path-Following Under Unknown Ocean Currents with Limited Data

Xinmao Li [1,2], Lingbo Geng [1,*], Kaizhou Liu [1] and Yifeng Zhao [1,2]

[1] State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China; lixinmao@sia.cn (X.L.); liukzh@sia.cn (K.L.); zhaoyifeng@sia.cn (Y.Z.)
[2] University of Chinese Academy of Sciences, Beijing 100049, China
* Correspondence: genglingbo@sia.cn

Abstract: Minimizing experimental data while maintaining good AUV path-following performance is essential to reduce controller design costs and ensure AUV safety, particularly in complex and dynamic underwater environments with unknown ocean currents. To address this, we propose a conservative offline model-based Q-learning (CMQL) algorithm. This algorithm is robust to unknown disturbance and efficient in data utilization. The CMQL-based controller is trained offline with dynamics and kinematics models constructed from limited AUV motion data and requires no additional fine-tuning for deployment. These models, constructed by improved conditional neural processes, enable accurate long-term motion state predictions within the data distribution. Additionally, the carefully designed state space, action space, reward function, and domain randomization ensure strong generalization and disturbance rejection without extra compensation. Simulation results demonstrate that CMQL achieves effective path-following under unknown ocean currents with a limited dataset of only 1000 data points. This method also achieves zero-shot transfer, demonstrating its generalization and potential for real-world applications.

Keywords: autonomous underwater vehicle; path-following control; model-based offline reinforcement learning; conditional neural processes; limited data

Academic Editor: Mostafa Hassanalian

Received: 10 January 2025
Revised: 7 March 2025
Accepted: 11 March 2025
Published: 12 March 2025

Citation: Li, X.; Geng, L.; Liu, K.; Zhao, Y. Model-Based Offline Reinforcement Learning for AUV Path-Following Under Unknown Ocean Currents with Limited Data. *Drones* **2025**, *9*, 201. https://doi.org/10.3390/drones9030201

1. Introduction

AUV path-following control is crucial for many missions, such as docking [1], underwater exploration [2], and sea creature sampling [3]. As these research areas advance, performance requirements for AUV path-following will become stricter, especially in dynamic and complex underwater environments with unknown ocean currents. Most controllers demand considerable experimental effort and large-scale AUV motion data collection to compute accurate mathematical models or tune their parameters. However, these requirements are both difficult and unsafe to fulfill in underwater environments. Therefore, in this paper, we propose a novel control strategy for AUV path-following in the presence of unknown ocean currents, aimed at achieving effective control performance and disturbance rejection with limited AUV motion data.

Various control methods for achieving path-following have been documented in the literature. For model-free controllers, proportional–integral–derivative control (PID) [4] has been successfully applied to AUVs, but it fails to respond in real time to unknown ocean current variations. ADRC [5] can compensate for disturbances and uncertainties in real time, but it may require extensive experimentation to carefully adjust the parameters. For

model-based controllers, including sliding model control [6] and model predicative control (MPC) [7], accurate AUV mathematical models are fundamental. Although these models enhance data efficiency and reduce experimental requirements, accurately identifying their dynamic parameters remains a data-intensive and technically complex challenge.

Reinforcement learning (RL), compared to the aforementioned methods, is better suited for complex and dynamic environments, offering excellent performance in high-uncertainty scenarios. As an intelligent algorithm that learns from experimental data, RL has inspired numerous control schemes for AUV path-following applications. For example, Wang et al. [8] proposed an adaptive PID controller based on soft actor–critic (SAC) [9] for AUV path-following. By combining PID with RL, the controller retains a certain degree of interpretability, making it easier to understand and deploy in practical applications. The neural network model-based RL control method proposed by Ma. D. F et al. [10] offers the advantages of enhancing adaptability to complex dynamics and improving path-following accuracy in different environments. Fan et al. [11] presented an improved twin delayed deep deterministic policy gradient (TD3) algorithm for AUV path-following control, enhancing the traditional TD3 approach to improve convergence and stability in underwater environments. These studies demonstrate that RL has the potential to overcome unknown ocean currents. RL can also be categorized into model-free and model-based approaches, each with distinct challenges. Model-free RL depends on continuous trial-and-error training in real-world environments, whereas model-based RL utilizes environmental models derived from extensive experimental data. This reliance on substantial data and real-world interactions is a key reason why RL struggles to fully realize its potential in AUV control.

Model-based offline reinforcement learning (MORL) is key to reducing experimental data and making RL more practical for AUV control. This method, with its lower accuracy requirements for models, can leverage offline training methods to further improve data efficiency and training safety. As a type of offline reinforcement learning (ORL), MORL generates a large amount of synthetic experience data by constructing an environment model from offline experimental data, reducing the need for offline experimental data. Additionally, the offline training method ensures the safety of the AUV. ORL has been applied in various fields, such as healthcare [12], autonomous driving [13], and robotics [14]. The challenge of ORL arises from the distribution shift between the offline dataset and the learned policy during training. It is mainly addressed by incorporating conservatism or regularization into standard RL algorithms. For example, Ma. C. Z. et al. [15] use in-sample value estimation to compute the advantage function for state–action pairs, and they incorporate a behavior cloning regularization term during the policy update. It has been validated across multiple tasks based on a real manipulator. Guan et al. [16] use environmental dynamics uncertainty to eliminate unknown actions in Q-value evaluation and combine trajectory information with Gaussian noise to increase the probability of optimal actions. Rafailov et al. [17] propose learning a latent-state dynamics model with uncertainty representation in the latent space, maximizing a lower bound of the evidence lower bound in unknown partially observable Markov decision processes. As another type of ORL, model-free ORL [18–20] requires substantial offline experimental data to ensure policy performance, yet its offline training techniques remain valuable. MORL [21–23] improves data efficiency by leveraging model learning techniques, such as maximum likelihood estimation [23], Gaussian process [24], local linear model [25], and neural network [26]. However, the offline experimental data required by these modeling techniques are still quite large in underwater environments. Therefore, developing a high-precision AUV modeling method based on limited data is crucial.

The above research clearly shows that MORL offers a novel paradigm for underwater missions, such as AUV path-following. By reducing the offline experimental data required, its practical value in underwater missions can be significantly enhanced. Therefore, this paper proposes a conservative offline model-based Q-learning algorithm (CMQL), consisting of model learning and policy learning. For model learning, we propose a novel AUV modeling method using limited data, relying on the conditional neural process (CNP) method [27]. It enables accurate long-term prediction, significantly reducing the AUV offline motion data required. For policy learning, the regularization technique of CQL [20] is then applied within the SAC algorithm to make policy training more conservative, addressing distribution shifts and model biases. This enables the CMQL-based controllers to achieve zero-shot transfer to the actual AUV. In addition, to ensure AUV path-following performance under unknown ocean currents, a two-stage RL-based controller structure is meticulously designed. Domain randomization (DR) is used during training to further enhance the controllers' disturbance rejection of unknown ocean currents.

The organization of the rest of this paper is as follows: Section 2 provides a brief introduction to the under-actuated AUV model and presents the path-following problem to be addressed. Section 3 offers a detailed description of the CNP-based AUV modeling method, along with the improved training method to enhance the model's long-term prediction accuracy. Section 4 outlines the controller design scheme, including the modeling of the AUV path-following problem as a Markov decision process (MDP), the CMQL algorithm, and the training method using domain randomization. In Section 5, the controller's performance is validated through a simulation. Finally, Section 6 offers a brief summary of the research.

2. Preliminaries

2.1. Model of the Under-Actuated AUV

This article focuses on a torpedo-shaped under-actuated AUV with a cross-rudder and single-propeller actuator layout. The AUV motion under ocean currents is described using an earth-fixed frame $\{E\}$ and a body-fixed frame $\{B\}$, as shown in Figure 1. Thus, the kinematics and dynamics of this type of AUV are given as follows:

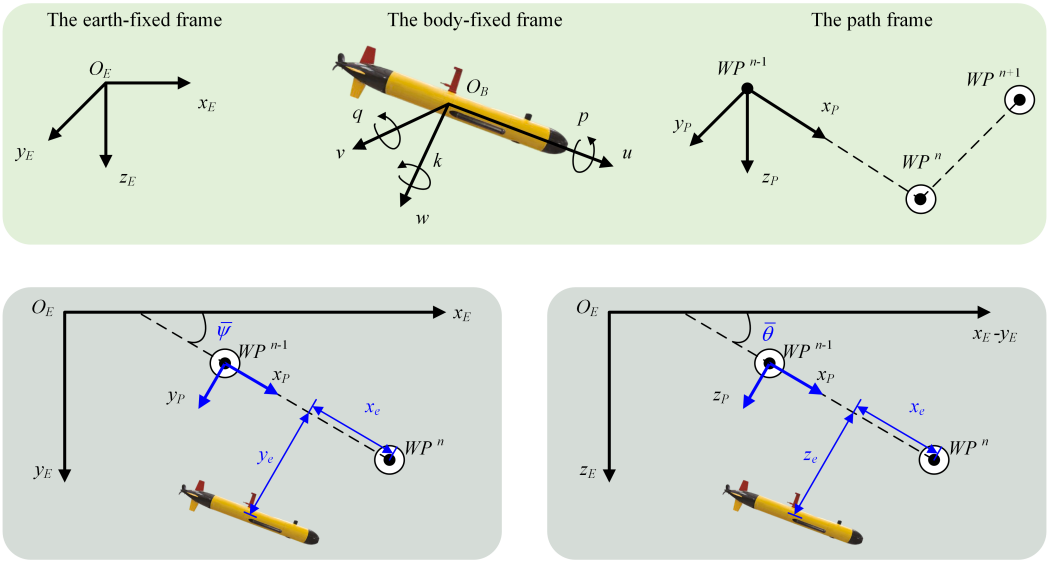

Figure 1. Illustration of the path-following mission of AUV.

$$\dot{\boldsymbol{\eta}} = \boldsymbol{T}(\boldsymbol{\eta})\boldsymbol{v}_r + \boldsymbol{V}_c \tag{1}$$

$$\boldsymbol{M}\dot{\boldsymbol{v}}_r + \boldsymbol{C}(\boldsymbol{v}_r)\boldsymbol{v}_r + \boldsymbol{D}(\boldsymbol{v}_r)\boldsymbol{v}_r = \boldsymbol{G}(\boldsymbol{v}_r)\boldsymbol{\tau} \tag{2}$$

where, in the $\{E\}$, $\boldsymbol{\eta} = [x, y, z, \varphi, \theta, \psi]^T$ represents the position and the orientation of AUV, respectively. $\boldsymbol{V}_c = [V_c^x, V_c^y, 0, 0, 0, 0]^T$ is the ocean current in $\{E\}$, and $\boldsymbol{v}_c = [u_c, v_c, w_c, 0, 0, 0]^T$ is the ocean current in $\{B\}$. $\boldsymbol{V}_c = \boldsymbol{T}(\boldsymbol{\eta})\boldsymbol{v}_c$, $\boldsymbol{T}(\boldsymbol{\eta})$ is the transformation matrix. In the $\{B\}$, $\boldsymbol{v} = [u, v, w, p, q, k]^T$ is the velocity, and $\boldsymbol{\tau} = [\tau_x, \tau_r, \tau_s]^T$ is the control signal, where τ_x, τ_r andτ_s denote the propeller thrust, vertical rudder angle and horizontal rudder angle, respectively. The velocity relative to the ocean current is denoted as $\boldsymbol{v}_r \triangleq \boldsymbol{v} - \boldsymbol{v}_c = [u_r, v_r, w_r, p, q, k]^T$. $\boldsymbol{M}$ is the inertia matrix, $\boldsymbol{C}(\boldsymbol{v})$ is the Coriolis and centripetal matrix, $\boldsymbol{C}(\boldsymbol{v})$ is the hydrodynamic damping matrix, and $\boldsymbol{C}(\boldsymbol{v})$ is the force and torque matrix of the actuator.

2.2. Problem Formulation

The path-following mission involves the AUV following a time-independent path under unknown ocean currents. This path is defined by a straight line connecting a series of successive waypoints $\mathcal{WP} \triangleq \{WP^n, n = 1, ..., N\}$, where $WP^n = (x_d^n, y_d^n, z_d^n)$, and WP^0 is the initial AUV location. We define a path frame $\{W\}$ with its origin at waypoint WP^{n-1} and the x-axis pointing towards waypoint WP^n. Therefore, the path-following error (x_e, y_e, z_e) in $\{W\}$ is computed as follows:

$$\begin{bmatrix} x_e \\ y_e \\ z_e \end{bmatrix} = \boldsymbol{T}_W^T \begin{bmatrix} x - x_d^n \\ y - y_d^n \\ z - z_d^n \end{bmatrix} \tag{3}$$

$$\boldsymbol{T}_W = \begin{bmatrix} \cos\bar{\theta}\cos\bar{\psi} & -\sin\bar{\psi} & \sin\bar{\theta}\cos\bar{\psi} \\ \cos\bar{\theta}\sin\bar{\psi} & \cos\bar{\psi} & \sin\bar{\theta}\sin\bar{\psi} \\ -\sin\bar{\theta} & 0 & \cos\bar{\theta} \end{bmatrix} \tag{4}$$

$$\left\{ \begin{array}{l} \bar{\psi} = \text{atan2}(y_d^n - y_d^{n-1}, x_d^n - x_d^{n-1}) \\ \bar{\theta} = \text{atan2}(z_d^n - z_d^{n-1}, \sqrt{(y_d^n - y_d^{n-1})^2 + (x_d^n - x_d^{n-1})^2}) \end{array} \right] \tag{5}$$

where atan2(a,b) returns the arc tangent of a/b within the bound $(-\pi, \pi]$. $\boldsymbol{T}_W$ is the transformation matrix. In the $\{W\}$, $\bar{\psi}$ is the angle between the vector $\overrightarrow{WP^{n-1}WP^n}$ and the y-axis; similarly, $\bar{\theta}$ is the angle with the z-axis. It is noteworthy that y_e is only related to the AUV's heading angle, while z_e is solely related to the pitch angle.

Therefore, under unknown ocean currents, the objective of this paper is to minimize the path-following errors y_e and z_e while maintaining the desired surge velocity u_d, as shown in Figure 1. Furthermore, we make the following assumptions: (1) The physical parameters of the AUV are completely unknown, and only a limited amount of motion data was collected from the environment in the absence of ocean current disturbances. (2) The focus is solely on the ocean current in the horizontal plane, which is unknown and unmeasurable.

We adopt the MORL approach to achieve this objective through two components: AUV model learning and controller designing, as detailed below.

(1) **AUV model learning:** To provide the controller with synthetic experience data under different ocean currents, AUV dynamics and kinematics models are constructed based on CNP. These models are required to predict the long-term motion states of the AUV with high precision and are trained using the limited data.

(2) **Controller designing:** For the path-following mission under unknown ocean currents, a two-stage controller structure is designed. In the first stage, the path-following controller is to convert path-following errors y_e and z_e into the desired yaw angular velocity k_d and pitch angular velocity q_d. In the second stage, three controllers are used: the yaw velocity controller and the pitch velocity controller,

which ensure that the angular velocity tracking errors $k - k_d$ and $q - q_d$ gradually tend to zero over time, and the surge velocity controller, which ensure that the actual surge velocity u matches the desired velocity u_d. These controllers are trained using CNP-based dynamics and kinematics models and can achieve zero-shot transfer to actual AUV.

3. AUV Model Learning

This paper utilizes the CNP [27] method to construct the AUV model, which serves as the interactive training environment for the offline training of the controller. To enable the randomization of ocean currents during RL policy training, we separately construct dynamics and kinematics models using the same method. These CNP-based models achieve good AUV state prediction accuracy by relying solely on a limited AUV motion dataset. Although high-precision prediction is only achieved for AUV states within the dataset distribution, it is sufficient to meet the requirements for RL policy training. Note that the roll ϕ of the under-actuated AUV is self-stabilizing, and the sway velocity v and the heave velocity w cannot be directly controlled. Therefore, the CNP-based dynamics model only predicts the surge velocity u, pitch angular velocity q, and yaw angular velocity k based on the control signal τ. The CNP-based kinematics model is used to predict the position increments $\dot{x}, \dot{y}, \dot{z}$ based on the surge velocity u, pitch angle θ, and heading angle ψ.

This section provides a detailed overview of the CNP-based AUV modeling method and the improved training techniques that enhance prediction accuracy.

3.1. Structure

The state of the AUV at the next time step is determined by the current state $\boldsymbol{sm}$ and the current action $\boldsymbol{am}$. Based on this, the AUV model is defined as a stochastic process. Consider a set $O = \{[(\boldsymbol{sm}_{t-1}, \boldsymbol{am}_{t-1}), \boldsymbol{sm}_t]\}$ of pairs of input $(\boldsymbol{sm}_{t-1}, \boldsymbol{am}_{t-1})$ and output $\boldsymbol{sm}_t$ and another set $T = \{(\boldsymbol{sm}_t, \boldsymbol{am}_t)\}$ of unlabeled points. These sets are called the set of observations and targets, respectively. For the CNP-based dynamics model, $\boldsymbol{sm} = [u, q, k]$ and $\boldsymbol{am} = \boldsymbol{\tau}$, while for the CNP-based kinematics model, $\boldsymbol{sm} = [\dot{x}, \dot{y}, \dot{z}]$ and $\boldsymbol{am} = [u, \theta, \psi]$. In the stochastic process, the probability distribution P over functions $f : (\boldsymbol{sm}_t, \boldsymbol{am}_t) \rightarrow \boldsymbol{sm}_{t+1}$ is defined. Then, for $f \sim P$, set $\boldsymbol{sm}_{t+1} = f(\boldsymbol{sm}_t, \boldsymbol{am}_t)$, and P defines a joint distribution over the random variables $\{f(\boldsymbol{sm}_t, \boldsymbol{am}_t)\}$. Therefore, there is a conditional distribution $P(f(T)|O, T)$. The objective is to predict the output $f(\boldsymbol{sm}_t, \boldsymbol{am}_t)$ for every $(\boldsymbol{sm}_t, \boldsymbol{am}_t) \in T$ given O. For this purpose, CNP parametrizes the distribution P over $f(T)$ as P_ϑ, and we map the observation set into an embedding of fixed dimensionality, as follows:

$$\rho = h_\vartheta((\boldsymbol{sm}_{t-1}, \boldsymbol{am}_{t-1}), \boldsymbol{sm}_t) \quad \forall((\boldsymbol{sm}_{t-1}, \boldsymbol{am}_{t-1}), \boldsymbol{sm}_t) \in O \tag{6}$$

where h_ϑ is a learnable encoder neural network. Since the observation set contains only a single data pair, there is no need to aggregate the output of the encoder network, as in the original CNP algorithm. Then, the encoded observation set is decoded along with the target set as follows:

$$\phi = \mathcal{G}_\vartheta((\boldsymbol{sm}_t, \boldsymbol{am}_t), \rho) \quad \forall(\boldsymbol{sm}_t, \boldsymbol{am}_t) \in T \tag{7}$$

where $\mathcal{G}_\vartheta$ is a learnable decoder neural network. ϕ is used to parameterize the mean and variance $\phi = (\mu, \sigma^2)$ of Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ for each $(\boldsymbol{sm}_t, \boldsymbol{am}_t) \in T$.

In short, as shown in Figure 2, CNP can be described as the following conditional distribution:

$$P_\vartheta(f(T)|O, T) = P_\vartheta(f(T)|\mathcal{G}_\vartheta(T, h_\vartheta(O))) \tag{8}$$

and can predict the state $\boldsymbol{sm}_{t+1}$ by inputs consisting of $\boldsymbol{sm}_t$, $\boldsymbol{am}_t$, $\boldsymbol{sm}_{t-1}$, $\boldsymbol{am}_{t-1}$.

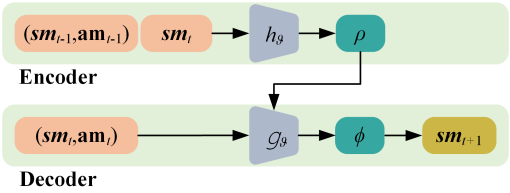

Figure 2. Schematic representation of the CNP.

3.2. Training

The collection of the training dataset employs practical and safe strategy. A coarsely tuned PID controller is employed for simultaneously tracking the surge velocity, heading angle, and pitch angle in order to collect the required motion data. This approach ensures the safety and real-time nature of AUV motion dataset collection. The desired targets for surge velocity, heading angle, and pitch angle are composite functions composed of sinusoids with different frequencies. Both the CNP-based dynamics and kinematics models rely on these data collection strategy. The diversity of the samples directly impacts the generalization of CNP-based AUV models.

To enable the model to achieve stable long-term prediction capability, we propose an N-step recurrent iterative training method, as illustrated in Figure 3 and Algorithm 1. In each iteration of training, the CNP performs N-step recurrent predictions. The predicted value $f(T_{n-1})$ at step $n-1$ is used as the input $\boldsymbol{sm}_n$ for the model at step n, and the action $\boldsymbol{am}$ is known at each step. The model is trained by randomly sampling subsets from the motion dataset, selecting observation and target splits, passing the data to this structure, computing the loss, and performing stochastic gradient updates until convergence. Its loss function is defined as follows:

$$\mathcal{L} = -\sum_{n=1}^{N} \log P_\vartheta(\boldsymbol{sm}_{n+1}|O_n, T_n) + \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{sm}_{n+1} - f(T_n))^2 \tag{9}$$

where the first term is the sum of the negative log-likelihood over N steps, while the second term is the N-step mean squared error between the true values and the predicted values.

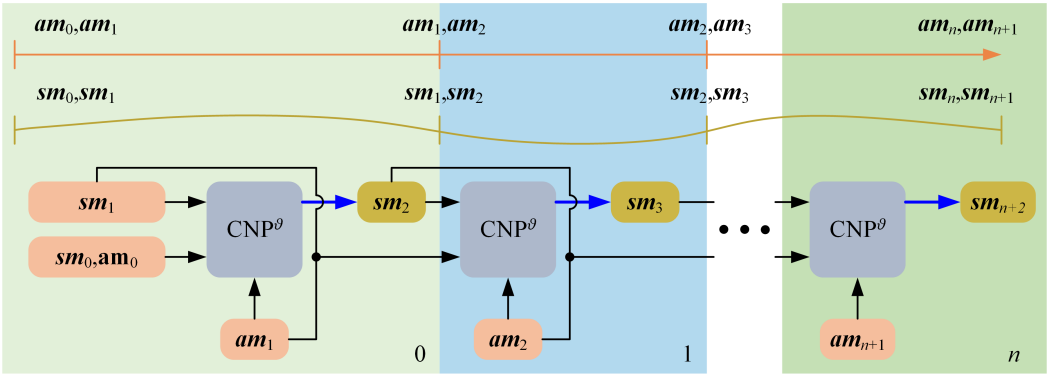

Figure 3. Schematic representation of the N-step recurrent iterative training method.

Algorithm 1 CNP multi-step iterative training algorithm.

Input: The limited offline dataset
Result: The trained CNP-based AUV models
Initialize the CNP network parameters ϑ
while not done **do**
  Sample *batch* from dataset
  Sample M samples $\{D_m\}_{m=1}^{M} \sim batch$, where $\{O_n, T_n\}_{n=1}^{N} \sim D_m$ is a consecutive set
  for $m = 1, \ldots, M$ **do**
    for $n = 1, \ldots, N$ **do**
      Compute the loss about $-\log P_\vartheta(\boldsymbol{sm}_{n+1}|O_n, T_n)$
      Compute the loss about $(\boldsymbol{sm}_{n+1} - f(T_n))^2$
      Make $\boldsymbol{sm}_{n+1} = f(T_n)$
    end for
    Compute the total loss by (9)
    Update network parameters ϑ
  end for
end while

4. Controller Designing

The two-stage controller structure is shown in Figure 4. All four controllers in this structure, including path-following, yaw angular velocity, pitch angular velocity, and surge velocity, are modeled similarly to the MDP. They are trained offline using the same policy optimization method, which encompasses conservative offline model-based Q-learning (CMQL). This method utilizes CNP-based AUV models, constructed from the limited motion dataset, as the interactive training environment and employs the regularization technique from CQL [20] for policy optimization. CMQL not only overcomes the reality gap caused by distribution shift and model bias in the CNP-based AUV models but also significantly reduces the required data compared to CQL. The controllers trained by CMQL can make zero-shot transfer from the CNP-based models to actual AUV. In this section, we provide a detailed description of the MDP design for the controllers, the CMQL method, and the training process of the controllers.

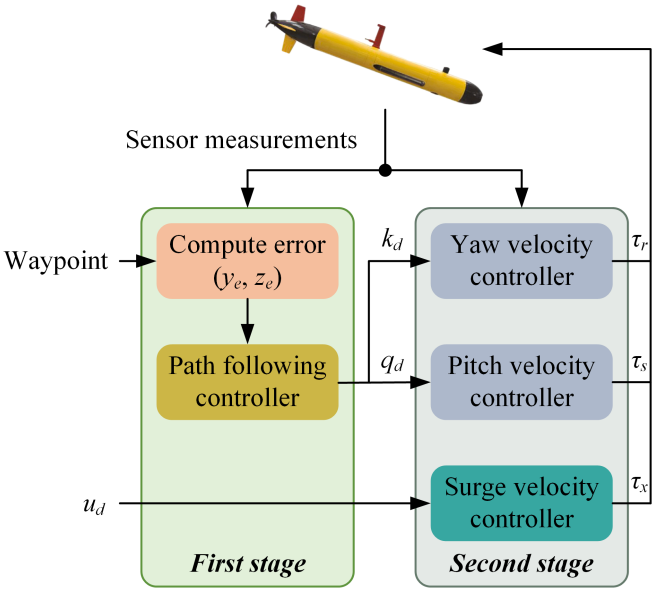

Figure 4. Overall structure of the proposed control.

4.1. MDP Modeling

The MDP is a mathematical framework of the RL problem, defined by the tuple. Here, S represents the state space, A denotes the action space, P is the state transition probability, and R is the reward function. We can describe the interaction process between the controller and the AUV motion environment using the MDP framework. At time t, the controller takes action $\boldsymbol{a}_t$ upon receiving the state $\boldsymbol{s}_t$. The state transfers to $\boldsymbol{s}_{t+1}$ with a probability $p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$, and the current policy of the controller is evaluated based on the immediate reward r_t. The objective of the RL problem is to find a policy π to maximize the long-term cumulative reward $J(\pi) = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\pi}[\sum_{t=0}^{T-1} \gamma^t r_t]$, where γ is the scalar discount factor.

Therefore, the definitions of the four components of the MDP are crucial for the performance of the path-following mission. To reduce the complexity of the algorithm, we adopt a similar scheme for the action space, state space, and reward function across these controllers.

Action space: As the output of the controllers, both the range of the value and its rate of change should be considered. Therefore, the action space is designed as follows:

$$\boldsymbol{a} = [\Delta k_d, \Delta q_d, \Delta\tau_x, \Delta\tau_r, \Delta\tau_s] \quad \boldsymbol{a} \in [-\boldsymbol{\zeta}, \boldsymbol{\zeta}] \tag{10}$$

where ζ is the bound of rate of change for these controllers' output. Note that the output of the controllers is $\Sigma\boldsymbol{a}_t = \boldsymbol{a}_t + \Sigma\boldsymbol{a}_{t-1} = [k_d, q_d, \tau_x, \tau_r, \tau_s], \Sigma\boldsymbol{a}_t \in [-\boldsymbol{\varsigma}, \boldsymbol{\varsigma}]$, where ς is the bound of range for these controllers' output. This action space is generalized, allowing different controllers to select various action parameters as needed. For example, the path-following controller is $\boldsymbol{a} = [\Delta k_d, \Delta q_d]$, and the surge velocity controller is $\boldsymbol{a} = [\Delta\tau_x]$.

State space: The state space represents the real-time state of the AUV during path-following. A more comprehensive state space leads to better controller performance. To ensure that the controller can overcome unknown ocean currents, the state distribution of the RL must remain unaffected by changes in the ocean currents. Therefore, three types of state vectors are designed within the entire state space.

The error state vector used to express the path-following errors and velocity tracking errors is as follows:

$$\boldsymbol{e}_t = \begin{cases} [y_{e,t}, z_{e,t}, \psi_{s,t} - \bar{\psi}, \theta_{s,t} - \bar{\theta}] & \textit{if path following controller} \\ [u_t - u_d, k_t - k_d, q_t - q_d] & \textit{others} \end{cases} \tag{11}$$

$$\begin{cases} \psi_{s,t} = \text{atan2}(y_t - y_{t-1}, x_t - x_{t-1}) \\ \theta_{s,t} = \text{atan2}(z_t - z_{t-1}, \sqrt{(y_t - y_{t-1})^2 + (x_t - x_{t-1})^2}) \end{cases} \tag{12}$$

Like the action space, different controllers correspond to different elements in (11). For the yaw velocity controller, it is $\boldsymbol{e}_t = [k_t - k_d]$. The error variation vector $\Delta\boldsymbol{e}$ is

$$\Delta\boldsymbol{e} = [\boldsymbol{e}_t - \boldsymbol{e}_{t-1}] \tag{13}$$

At last, the action space consists of the increments generated by each controller's output, which does not fully describe the state changes caused by these outputs. Therefore, the values of each output need to be included in the state space, as follows:

$$\boldsymbol{o} = \begin{cases} [k_d, k_d - k, q_d, q_d - q] & \text{if path following controller} \\ [\tau_x, \tau_r, \tau_s] & \text{others} \end{cases} \tag{14}$$

Additionally, $k_d - k$ and $q_d - q$ in the path-following controller are used to monitor the tracking of yaw angular velocity and pitch angular velocity, ensuring that the outputs

are reasonable. Consequently, the state space is defined by combining the vectors above as follows:

$$\boldsymbol{s} = [\boldsymbol{e}_t, \Delta\boldsymbol{e}, \boldsymbol{o}] \tag{15}$$

Reward function: The reward function r guides the direction of policy updates, and its design depends on the MDP's state and the mission requirements. Based on the path-following objective described in Section 2, the one-step reward function is designed as follows:

$$r = -\omega_e \|\boldsymbol{e}_t\|_1 - \omega_\Delta \|\Delta\boldsymbol{e}\|_1 - \omega_a \|\Sigma\boldsymbol{a}\| \tag{16}$$

where $\omega_i, i = e, \Delta, a$ are weight coefficients. $\|(\cdot)\|_1$ denotes the 1-norm of $(\cdot)$. The first two terms guide the state errors to zero, while the last term encourages smaller outputs from the controllers to minimize energy consumption.

4.2. Conservative Offline Model-Based Q-Learning

The CMQL method used in this article, as shown in Algorithm 2, is based on the SAC method [9], and the objective of the RL problem can be formulated as

$$\max_\pi J(\pi) = \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{a}_t)\sim\pi}\left[\sum_{t=0}^{T} \gamma^t (r_t + \alpha H(\pi(\cdot|\boldsymbol{s}_t)))\right] \tag{17}$$

where α is the temperature parameter and $H(\pi(\cdot|\boldsymbol{s}_t)$ is the entropy term. Unlike standard RL, this paper interacts with the CNP-based AUV model learned from a limited motion dataset. The policy is optimized based solely on the model-based dataset $D_\mathcal{M}$, which is generated by the CNP-based AUV models. The CNP-based AUV models achieve high state prediction accuracy within a limited range. However, it is difficult to confine policy exploration entirely within this range during training, and overly limited exploration can adversely affect policy performance. This is primarily due to the limited amount of AUV motion data, which results from the challenges of collecting data in underwater environments. The distribution shift and model bias between the CNP-based AUV model and the actual environment can lead to the failure of standard RL methods in offline policy optimization. To address this issue, we apply the regularization technique of CQL to SAC, obtaining a conservative policy π by learning the lower bound of the true action-value function Q^π. The action-value function Q^π of SAC is defined as follows:

$$Q^\pi(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{a}_t)\sim\pi}\left[\sum_{t=0}^{T} \gamma^t (r_t + \alpha H(\pi(\cdot|\boldsymbol{s}_t)))|\boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}\right] \tag{18}$$

and it is learned by iteratively applying the Bellman operator, which is expressed as follows:

$$\mathcal{B}^\pi Q^\pi(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}_{(\boldsymbol{s}, \boldsymbol{a})\sim\pi}\left[r + \gamma\left(Q^\pi(\boldsymbol{s}, \boldsymbol{a}) + \alpha H\left(\pi\left(\cdot|\boldsymbol{s}'\right)\right)\right)\right] \tag{19}$$

The network architecture of SAC employs an actor–critic framework, optimizing the policy by alternately updating the critic network Q_ϕ and the policy network π_ϕ. We incorporate a regularization term into the critic loss, which penalizes the action value function Q^π at states in the model-based dataset $D_\mathcal{M}$ for actions not observed in the model-based dataset $D_\mathcal{M}$. This enables a conservative estimation for the policy π, mitigating the challenges of model bias and distribution shift. The critic loss is expressed as follows:

$$\begin{aligned}\mathcal{L}_{\text{critic}} &= \beta\left(\mathbb{E}_{\boldsymbol{s}\sim D_\mathcal{M}, \boldsymbol{a}\sim\mu(\cdot|\boldsymbol{s})}\left[Q_\phi(\boldsymbol{s}, \boldsymbol{a})\right] - \mathbb{E}_{\boldsymbol{s}, \boldsymbol{a}\sim D_\mathcal{M}}\left[Q_\phi(\boldsymbol{s}, \boldsymbol{a})\right]\right) \\ &+ \frac{1}{2}\mathbb{E}_{(\boldsymbol{s}, \boldsymbol{s}, \boldsymbol{s}')\sim D_\mathcal{M}}\left[\left(Q_\phi(\boldsymbol{s}, \boldsymbol{a}) - \mathcal{B}^\pi Q^\pi(\boldsymbol{s}, \boldsymbol{a})\right)^2\right]\end{aligned} \tag{20}$$

where β is the conservative coefficient used to balance the level of conservatism in Q^{π}. $\mu(\cdot|s)$ is a uniform distribution over the action bound. The policy network π_{ϕ} is updated based on the following loss function:

$$\mathcal{L}_{policy} = \mathbb{E}_{\boldsymbol{s}\sim D_{\mathcal{M}}, \boldsymbol{a}\sim\pi_{\phi}}\left[-Q_{\phi}(\boldsymbol{s},\boldsymbol{a}) - \alpha H\left(\pi_{\phi}(\cdot|\boldsymbol{s})\right)\right] \tag{21}$$

Algorithm 2 Conservative offline model-based Q-learning.

1: **Input:** The CNP-based AUV models
2: **Result:** Conservative policy π
3: Initialize parameters of the policy network π and the critic network Q
4: Initialize the replay buffer $D_{\mathcal{M}}$
5: **for** each episode **do**
6: Collect data with π, add data to $D_{\mathcal{M}}$
7: Update Q by repeatedly solving (20) using samples from $D_{\mathcal{M}}$
8: Update π by solving (21)
9: **end for**

4.3. Training

As described in Section 2.2, the control structure consists of a path-following controller and three velocity controllers. First, the surge velocity, yaw velocity, and pitch velocity controllers are independently trained based on the CNP-based dynamics model, as shown in Figure 5. Next, the control policy learning process for the path-following controller is illustrated in Figure 6. During training, the CNP-based dynamics and kinematics models are combined to form an interactive training environment, as shown in (22). Additionally, the trained yaw velocity and pitch velocity controllers are incorporated into the training process, generating the values of two rudder angles as inputs to the interactive environment based on the output of the path-following controller. The desired target for the surge velocity controller is provided separately.

Considering that the CNP-based AUV models may not fully reflect the AUV's motion, and considering that it is unrealistic to collect a comprehensive motion dataset, the desired target during training should remain within the range of the offline dataset. Once the trained policy is deployed on the AUV, this limitation is canceled. Additionally, it is crucial to expose the policy to different ocean current disturbances during the training process. Therefore, to enable the offline-trained controller to be deployed on the AUV without fine-tuning, we employed DR during training.

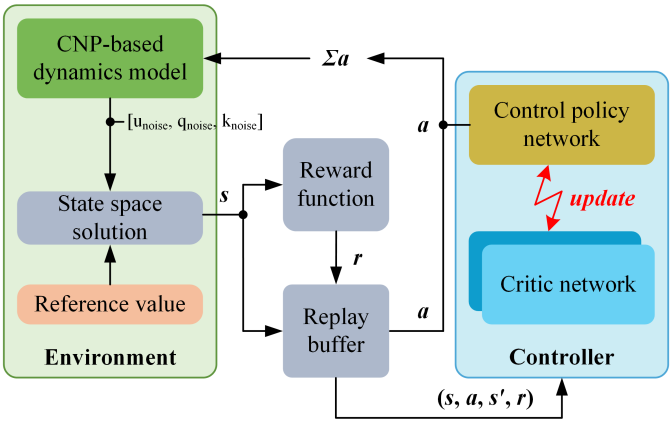

Figure 5. Control policy learning schematic representation of the velocity controllers.

DR can enhance the disturbance rejection of RL-based controllers ensuring that they perform well in environments with unknown ocean currents. Specifically, it involves

randomizing multiple parameters across the training environment, including introducing random offsets u_{noise}, q_{noise}, k_{noise} in CNP-based dynamics model outputs to simulate unknown external force interference. Random settings of ocean currents $\boldsymbol{V}_c$ are introduced to diversify the training environment. Consequently, the formula for the training environment is expressed as follows:

$$\begin{cases} [\dot{x}, \dot{y}, \dot{z}]_{t+1} = f_{CNP-Kine}([\dot{x}, \dot{y}, \dot{z}]_t, [u, \theta, \psi]_{t+1}) + \boldsymbol{V}_c \\ [u, \dot{\theta}, \dot{\psi}]_{t+1} = f_{CNP-dyn}([u, q, k]_t, \tau_t) + [u_{noise}, q_{noise}, k_{noise}] \end{cases} \tag{22}$$

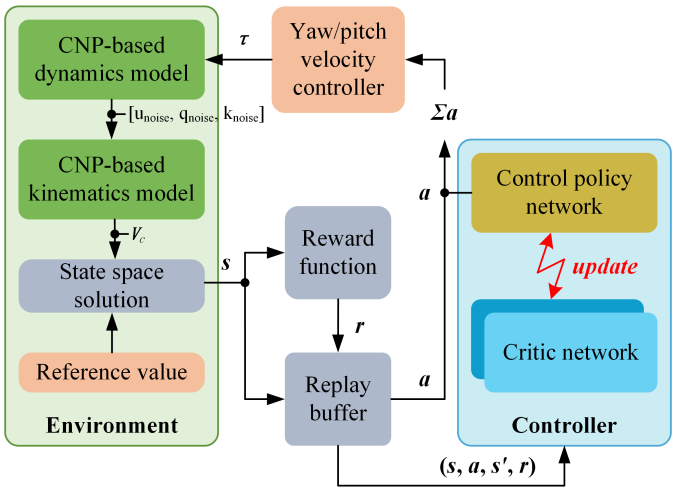

Figure 6. Control policy learning schematic representation of the path-following controller.

5. Simulation

In this section, we first validate the CNP-based modeling method for the AUV through simulation, followed by an evaluation of the proposed AUV path-following control method. It is important to note that the simulation environment uses a mathematical model of the self-designed under-actuated AUV, which is 7.985 m long and has a diameter of 0.40 m. The data sampling and control frequencies are both set to 2 Hz. The simulation is run on a laptop with an NVIDIA GeForce RTX 4060 GPU and an Intel i9 CPU.

5.1. CNP-Based AUV Model

We collected an offline dataset of 1000 samples to train the CNP-based AUV models. The network parameters of the CNP-based dynamics and kinematics models are identical. CNP was used to construct the two models consists of an encoder and a decoder, both based on a multilayer perceptron architecture. The encoder network has four hidden layers, while the decoder has three. Each hidden layer in both networks contains 512 neurons. The learning rate is set to 10^4, and the linear rectification function is used as an activation function. The model is trained for 3000 iterations.

We investigate the effect of the number of recurrent prediction steps per iteration on the model's convergence. As shown in Figure 7, increasing the number of prediction steps can improve the model's performance, but too many steps lead to a higher prediction error. We find that the optimal number of steps is 15 for the CNP-based dynamics model and 13 for the CNP-based kinematics model. To evaluate model performance, we use the coefficient of determination R^2, which measures the proportion of variance in the predictions relative to the total variance and indicates the correlation between the predicted and actual values, as shown in (23). We compare the true and predicted values of both models on a test set consisting of 200 consecutive samples, as shown in Figure 8. The two CNP-based models demonstrate strong capabilities in accurately reconstructing the AUV model. As shown

in Table 1, both models achieve an R^2 greater than 0.97, indicating strong generalization and the ability to predict unknown states accurately. This method significantly reduces the training data while maintaining high-precision prediction capabilities, offering a distinct advantage over other modeling approaches. However, due to the limited dataset, prediction accuracy for states outside the training dataset distribution may decrease.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} \tag{23}$$

where y_i is the actual value, $\hat{y}_i$ is the predicted value by model, and $\bar{y}_i$ is the mean of the actual values.

Table 1. The performance evaluation for the CNP-based dynamics and kinematics models.

	$\dot{x}$	$\dot{y}$	$\dot{z}$	u	q	k
R^2	0.9811	0.9734	0.9926	0.9933	0.9911	0.9952

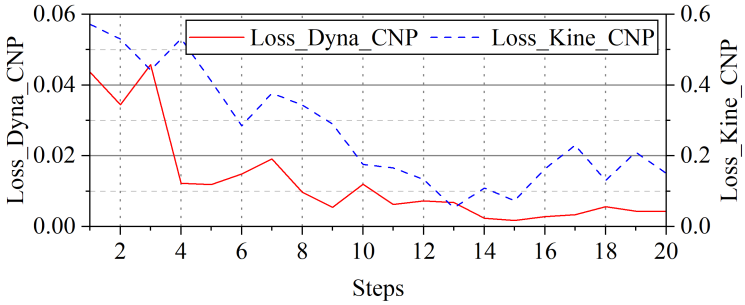

Figure 7. Impact of the recurrent prediction steps during each iteration on convergence.

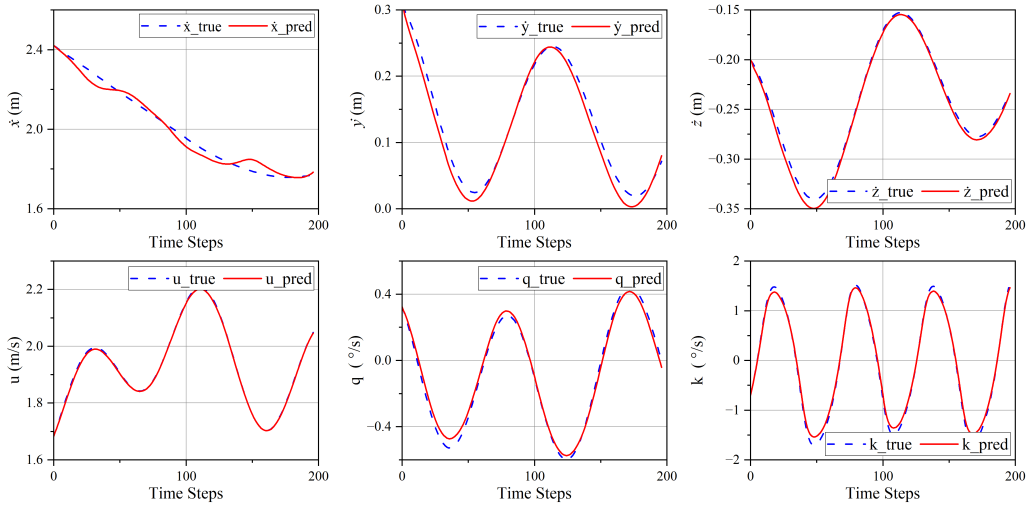

Figure 8. Prediction results of AUV motion states by the CNP-based dynamics and kinematics models.

5.2. Control Verification

In this subsection, we compare the proposed algorithm CMQL with different algorithms to validate its control performance and advantage during several controllers. Firstly, using the surge velocity controller, we compare CMQL with original CQL [20] and COMBO [23] to demonstrate the effectiveness of the CNP-based model. Next, we evaluated the control performance and generalization ability of CMQL in yaw velocity control under both disturbance-free and ocean current conditions. It is compared with SAC

trained in the simulation environment (SACSIM), to confirm whether the offline-trained CMQL could achieve the same performance as the online-trained algorithm. In the pitch velocity control, we validate the performance of CMQL and SAC trained with CNP-based dynamics model (SACCNP). Additionally, we construct a suboptimal CNP-based dynamics model and compare the control performance of CMQL and SAC using this model to assess the impact of model accuracy. These controllers, trained with the suboptimal model, are denoted as CMQL-sub and SACCNP-sub. Finally, in the complete path-following mission under both disturbance-free and ocean current disturbance conditions, we compare CMQL with MPC [28] based on the CNP-based models to verify whether CMQL can achieve good path-following control outside the training data distribution. MPC relies entirely on the accuracy of the model's predictions, which helps determine if there is any degradation in the prediction performance of the CNP-based model.

All RL controllers share the same network architecture and MDP design. Both the control policy and critic networks consist of fully connected layers, each with 256 nodes and 2 layers. The learning rates for the two networks are 10^{-5} and 3×10^{-4}, respectively. The configuration of various parameters for the domain randomization strategy during training is as follows: $u_{noise} \in [-0.1, 0.1]$ m/s, $q_{noise} \in [-0.1, 0.1]^\circ$, $k_{noise} \in [-0.1, 0.1]^\circ$, $V_c^x \in [-0.5, 0.5]$ m/s and $V_c^y \in [-0.5, 0.5]$ m/s. These parameters are randomly chosen at the start of each episode.

5.2.1. Surge Velocity Controller

We implement controllers using CMQL, CQL, and COMBO, respectively. All RL controllers' training lasts for 300 episodes. For CMQL, each episode contains 200 time steps, and the AUV's parameters are initialized to zero, with no control applied to τ_r and τ_s. The desired surge velocity is randomly selected within the range $u_d \in [1, 2]$ m/s for each training episode. For CQL and COMBO, they use an offline dataset of 2×10^5 samples. To illustrate the training process of CMQL, CQL and COMBO, we evaluate the controller in the simulation environment every five episodes.

The cumulative reward curve, which is shown in Figure 9, reflects the training process of the controller. CMQL and CQL take approximately 0.31 h, 1.7 h and 2.4 h to train, respectively. CQL and COMBO use a large dataset that is impractical to collect in reality. Despite requiring five times the training time of CMQL, their cumulative rewards still lag behind that of CMQL. The offline-trained controllers are directly tested in the simulation environment. The surge velocity control tracking results can be seen in Figure 10 and Table 2, with the desired surge velocity given in (24). CMQL shows more excellent control performance than CQL and COMBO. This indicates that, compared to modeling methods in other MORL, the CNP-based model can generate high-quality data for controller training with a limited offline dataset. Additionally, CMQL exhibits strong zero-shot transfer capability.

$$u_d = \begin{cases} 1 & t \leq 25 \\ 1.5 & 25 < t \leq 50 \\ 1 + 0.5\sin(0.025\pi t) & 50 < t \leq 100 \end{cases} \tag{24}$$

Table 2. The performance indexes for the surge velocity control.

	Setting Time (s)		**Mean Error (m/s)**		**Max Error (m/s)**
	0 s∼25 s	**25 s∼50 s**	**0 s∼25 s**	**25 s∼50 s**	**50 s∼100 s**
CQL	12	**6**	0.045	**−0.002**	0.081
COMBO	**10**	**6**	**0.001**	0.026	0.051
CMQL	**10**	10	**−0.003**	−0.005	**−0.012**

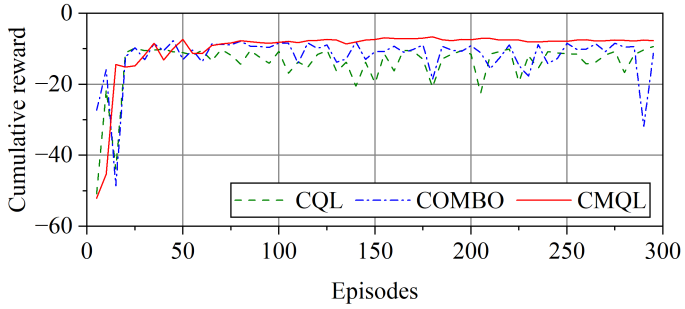

Figure 9. Cumulative reward curve of surge velocity controllers.

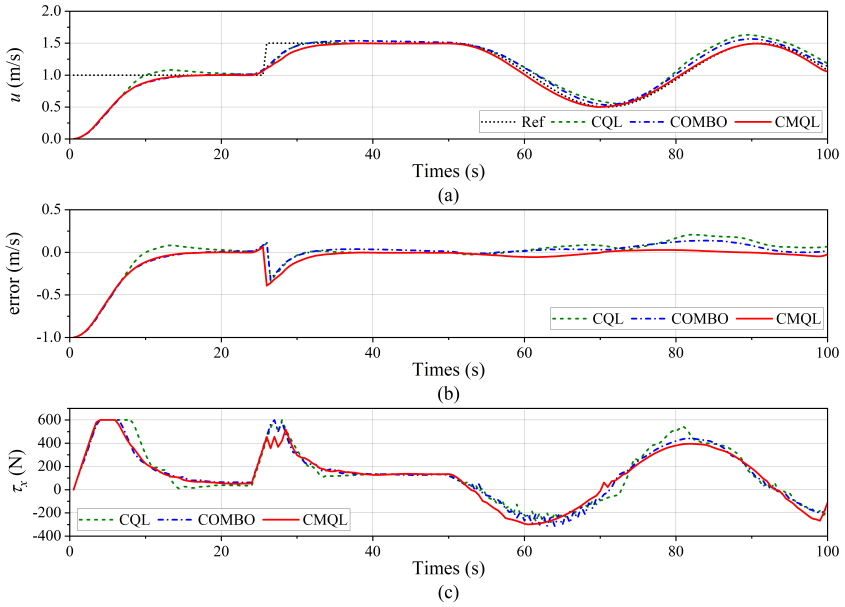

Figure 10. Tracking result of the surge velocity control. (**a**) Surge velocity tracking curve. (**b**) Tracking error. (**c**) Propeller thrust.

5.2.2. Yaw Velocity Controller

CMQL is similar in performance to SACSIM regarding yaw velocity control. Additionally, we trained a surge velocity controller based on SACSIM using the settings outlined in Section 5.2.1 for control validation. During training, the desired yaw velocity is randomly selected within the range $k_d \in [-0.5, 0.5]^\circ$, and no control is applied to τ_s. All other settings are the same as those for the surge velocity controller.

Both CMQL and SACSIM take approximately 0.3 h to train, with the results being illustrated in Figure 11. Compared to SACSIM, CMQL converges to the optimal parameters more quickly. We performed yaw velocity tracking under variable surge velocity, with the desired target defined in (25). Tests were conducted under both disturbance-free and ocean current conditions, with the ocean current set as $[V_c^x, V_c^y] = [0.5, -0.5]$ m/s. The results are presented in Figures 12–15 and Tables 3 and 4. Figures 12 and 13 show that CMQL with the CNP-based dynamics model achieves a tracking error comparable to that of SACSIM trained directly in the simulation environment. In the presence of ocean current disturbances, the tracking error of CMQL is even lower than that of SACSIM, as shown in Figures 14 and 15. This demonstrates that CMQL is a highly effective and safe method, allowing the controller to maintain excellent control performance and strong generalization. Its disturbance rejection capability surpasses that of RL controllers trained online or those based on the AUV's mathematical model.

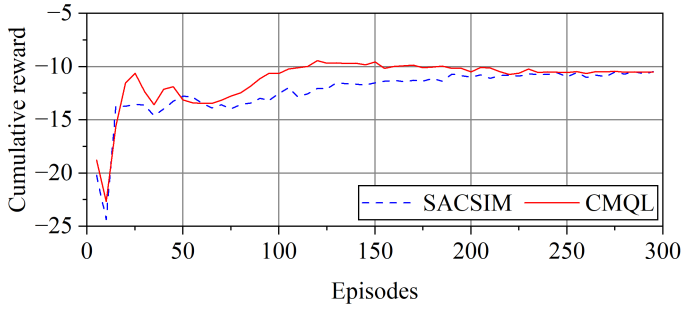

Figure 11. Cumulative reward curve of the yaw velocity controllers.

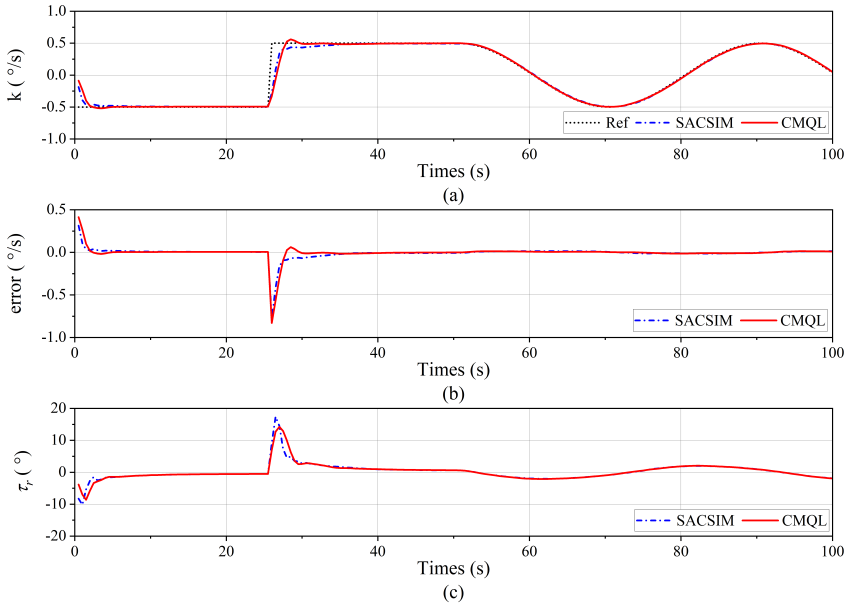

Figure 12. Yaw velocity tracking result of the yaw velocity control without disturbance. (**a**) Yaw velocity tracking curve. (**b**) Tracking error. (**c**) Vertical rudder angle.

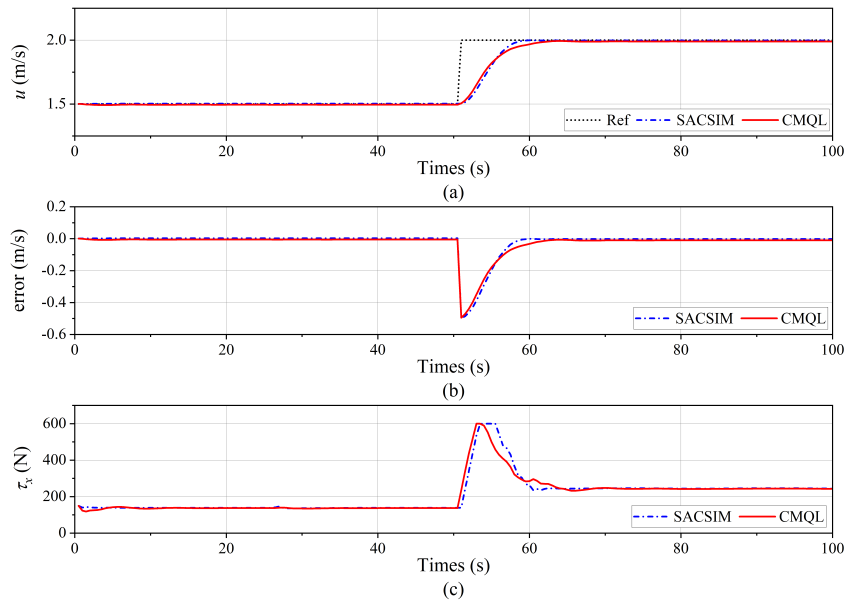

Figure 13. Surge velocity tracking result of the yaw velocity control without disturbance. (**a**) Surge velocity tracking curve. (**b**) Tracking error. (**c**) Propeller thrust.

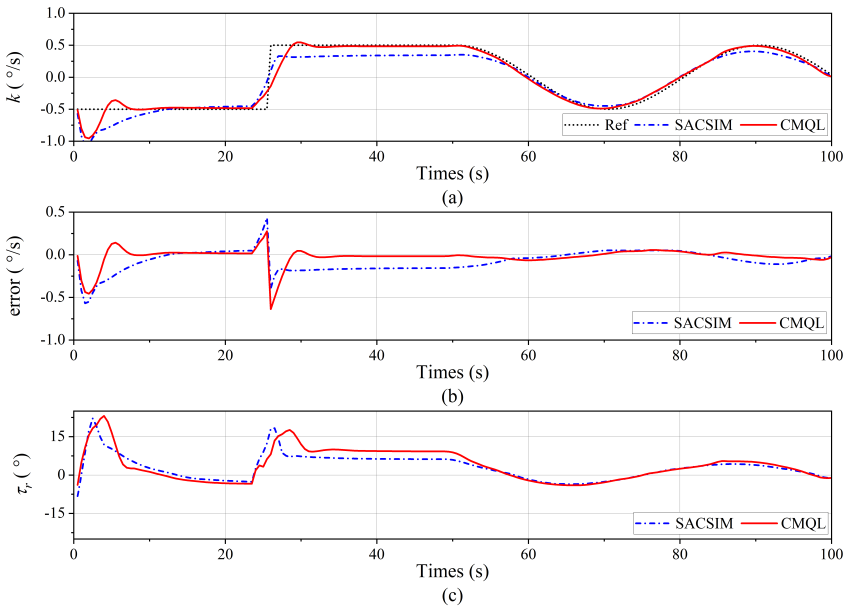

Figure 14. Yaw velocity tracking result of the yaw velocity control under ocean current disturbance. (**a**) Yaw velocity tracking curve. (**b**) Tracking error. (**c**) Vertical rudder angle.

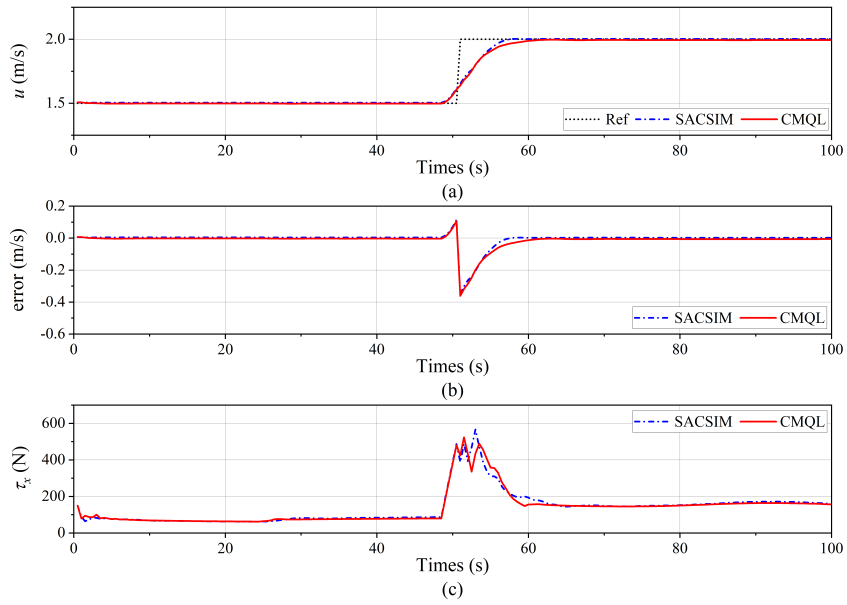

Figure 15. Surge velocity tracking result of the yaw velocity control under ocean current disturbance. (**a**) Surge velocity tracking curve. (**b**) Tracking error. (**c**) Propeller thrust.

Table 3. The yaw velocity tracking performance indexes for the yaw velocity control.

	Ocean Current (m/s)	**Setting Time (s)**		**Mean Error (°/s)**		**Max Error (°/s)**
	$[V_c^x, V_c^y]$	**0 s∼ 25 s**	**25 s∼ 50 s**	**0 s∼ 25 s**	**25 s∼ 50 s**	**50 s∼100 s**
SACSIM	$[0, 0]$	**2**	10	0.007	−0.009	0.002
	$[0.5, -0.5]$	14	**2**	0.035	−0.162	−0.027
CMQL	$[0, 0]$	2.5	**4**	**0.003**	**−0.007**	**0.001**
	$[0.5, -0.5]$	7	6	**0.018**	**−0.018**	**−0.009**

Table 4. The surge velocity tracking performance indexes for the yaw velocity control.

	Ocean Current (m/s)	**Setting Time (s)**	**Mean Error (°/s)**	
	$[V_c^x, V_c^y]$	**50 s∼100 s**	**0 s∼50 s**	**50 s∼100 s**
SACSIM	$[0, 0]$	**8**	**0.002**	$-$**0.003**
	$[0.5, -0.5]$	**6**	0.004	**0.002**
CMQL	$[0, 0]$	12	-0.005	-0.01
	$[0.5, -0.5]$	10	$-$**0.002**	-0.006

$$\begin{cases} k_d = \begin{cases} -0.5 & t \le 25 \\ 0.5 & 25 < t \le 50 \\ 0.5\sin(0.025\pi t) & 50 < t \le 100 \end{cases} \\ u_d = \begin{cases} 1.5 & t \le 50 \\ 2 & 50 < t \le 100 \end{cases} \end{cases} \tag{25}$$

5.2.3. Pitch Velocity Controller

For pitch velocity control, CMQL is compared with SACCNP, CMQL-sub, and SACCNP-sub. Additionally, the suboptimal CNP-based dynamics model is trained using a smaller offline dataset of 500 samples, with an R^2 for its predicted yaw velocity being less than 0.8. During training, the desired pitch velocity is randomly selected within the range $q_d \in [-0.5, 0.5]^\circ/\text{s}$, with $\tau_x = 300$ N and no control applied to τ_r.

The training times for CMQL, SACCNP, CMQL-sub, and SACCNP-sub are all approximately 0.3 h, with the results being shown in Figure 16. CMQL and SACCNP exhibit similar training outcomes. However, when trained on a suboptimal CNP-based dynamics model, CMQL-sub achieves a higher cumulative reward than SACCNP-sub. We conducted pitch velocity tracking under constant thrust, with the desired pitch velocity being defined in (26). The results are presented in Figure 17 and Table 5. The tracking errors for CMQL and SACCNP are minimal and similar, while the errors for CMQL-sub and SACCNP-sub increase but remain within an acceptable range. However, the pitch velocity controlled by SACCNP-sub exhibits oscillations caused by high-frequency, large-amplitude fluctuations of the horizontal rudder. The rudder oscillates by 8° every 0.5 s, which could potentially cause damage in real-world scenarios. This demonstrates that while SACCNP can achieve good control performance, it requires high prediction accuracy from the model. In contrast, CMQL, as a method for conservative policy optimization, can better adapt to models with a lower prediction accuracy. The policy optimization method of CMQL also reduces dataset requirements, making it more suitable for real-world applications.

$$q_d = \begin{cases} -0.5 & t \le 25 \\ 0.5 & 25 < t \le 50 \\ 0.5\sin(0.025\pi t) & 50 < t \le 100 \end{cases} \tag{26}$$

Table 5. The performance indexes for the pitch velocity control.

	Setting Time (s)		**Mean Error (°/s)**		**Max Error (°/s)**
	0 s∼25 s	**25 s∼50 s**	**0 s∼25 s**	**25 s∼50 s**	**50 s∼100 s**
SACCNP	**6**	5.5	$-$**0.004**	$-$**0.001**	$-$**0.113**
CMQL	9	**5**	0.005	-0.003	-0.132
SACCNP-sub	**12**	**5**	0.126	-0.086	0.210
CMQL-sub	13	**5**	**0.102**	$-$**0.054**	$-$**0.173**

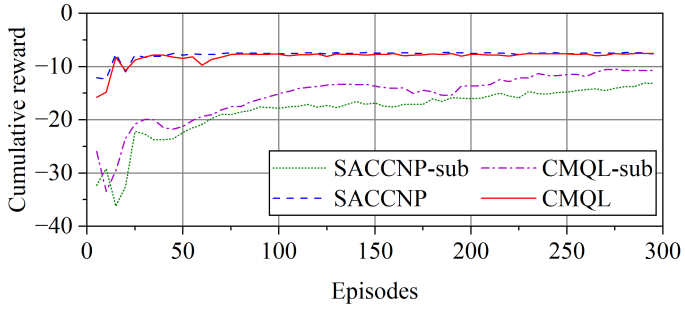

Figure 16. Cumulative reward curve of the pitch velocity controllers.

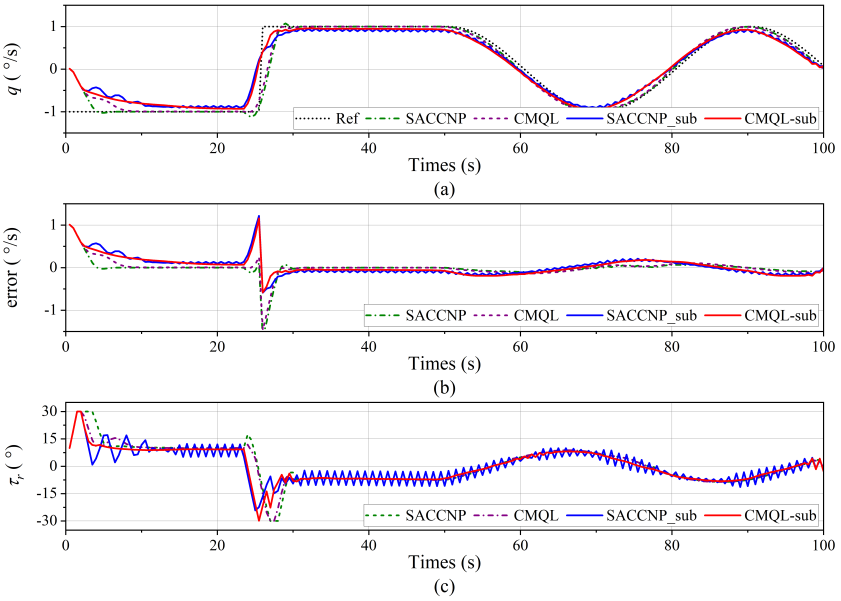

Figure 17. Pitch velocity tracking result of the pitch velocity control. (**a**) Pitch velocity tracking curve. (**b**) Tracking error. (**c**) Horizontal rudder angle.

5.2.4. Path-Following Mission

For the path-following mission, CMQL is compared with MPC based on CNP-based models. The training of CMQL lasts for 500 episodes, with each episode consisting of 300 time steps. All AUV parameters are initialized to zero, and the desired path points are defined as follows:

$$\begin{cases} x_d = \rho \cos(\omega) \\ y_d = \rho \sin(\omega) \end{cases} \tag{27}$$

where $\rho \in [200, 300]$ m and $\omega \in [-180, 180]^\circ$ are randomly selected within their respective ranges for each episode. All other settings are the same as those of the surge velocity controller. The training time for CMQL is approximately 0.5 h, with the training results being shown in Figure 18. MPC uses the same path-following control structure proposed in this paper. The control parameters of MPC are obtained using the particle swarm optimization algorithm. The path-following controller of MPC requires approximately 2 h, while each of the three MPC velocity controllers requiring about 0.5 h.

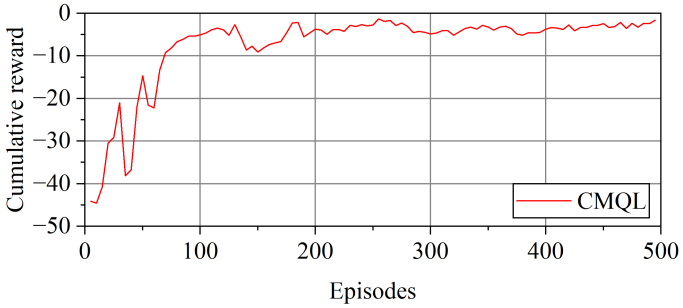

Figure 18. Cumulative reward curve of the path-following controller.

In the control simulation, we perform path-following missions on two different paths. The first is a polyline path consisting of the points [(0, 0, 0), (100, 5, 5), (170, 8, 5), (250, 15, 10), (420, 20, 10)]. On this path, we compare CMQL and MPC under both disturbance-free and ocean current disturbance conditions. The ocean current is defined as $[V_c^x, V_c^y] = [0.5, -0.5]$ m/s. The results are shown in Figures 19–22 and Table 6. Under disturbance-free conditions, the path-following errors for CMQL and MPC are small and similar, but MPC is less stable. Additionally, CMQL completes the task 4.5 s earlier than MPC. Under ocean current disturbances, CMQL still performs well and completes the task. In contrast, MPC fails because the CNP-based model cannot accurately predict the AUV's motion state under unknown ocean current disturbances. The conventional solution to this problem is to design an extended state observer to compensate for the controller's output, but this is difficult to achieve with a limited dataset. In contrast, CMQL uses the inaccurate generated data under ocean current disturbances and still achieves good disturbance rejection performance.

The second path is a spiral path defined by (28). We use CMQL to follow this path under ocean current disturbances, as defined in (29). The results are shown in Figures 23 and 24 and Table 7. The strong following performance reflects the generalization of CMQL. In Section 5.1, we mentioned that the CNP-based models cannot accurately predict the AUV's motion state outside the dataset distribution. Due to the limited number of samples, the dataset's distribution range is narrow. Using this model, CMQL still achieves unconstrained path-following under varying ocean currents after just 1400 episodes of training with four controllers. In contrast, MPC's control performance heavily depends on the accuracy of model predictions.

$$\begin{cases} x_d = 300\cos(0.002\pi n) - 300 \\ y_d = 300\sin(0.002\pi n) \\ z_d = 0.05n \end{cases} \tag{28}$$

$$\begin{cases} V_c^x = 0.5\sin(0.001\pi t) \\ V_c^y = 0.5\cos(0.001\pi t) \end{cases} \tag{29}$$

Table 6. The performance indexes for polyline path-following mission.

	Ocean Current (m/s)	Mean Error (m)		Max Error (m)		Completion
	$[V_c^x, V_c^y]$	y_e	z_e	y_e	z_e	Time (s)
MPC	$[0, 0]$	**0.010**	−0.103	0.598	−0.992	281.5
	$[0.5, -0.5]$	−5.972	−0.131	−51.449	−3.886	288
CMQL	$[0, 0]$	−0.025	**−0.089**	**0.502**	**−0.503**	**277**
	$[0.5, -0.5]$	**−0.242**	**−0.089**	**−4.621**	**−0.665**	**269.5**

Table 7. The performance indexes for spiral path-following mission under variable ocean current disturbance.

	Mean Error (m)		Max Error (m)	
	y_e	z_e	y_e	z_e
CMQL	−0.539	0.249	−0.873	0.356

In summary, the simulation results demonstrate that the proposed path-following method effectively accomplishes path-following missions under unknown ocean currents. This method is safe, efficient, and offers high control accuracy, strong generalization, and excellent engineering applicability.

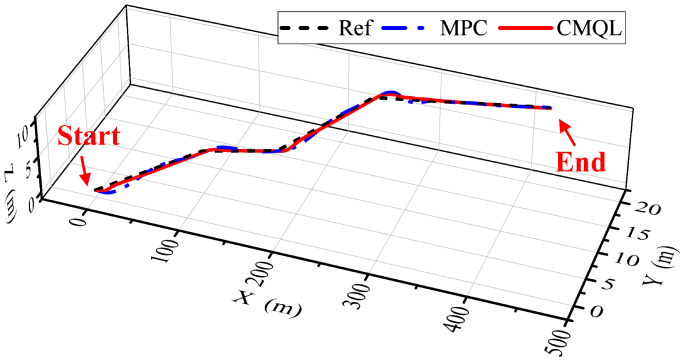

Figure 19. AUV trajectory of the polyline path-following mission without disturbance.

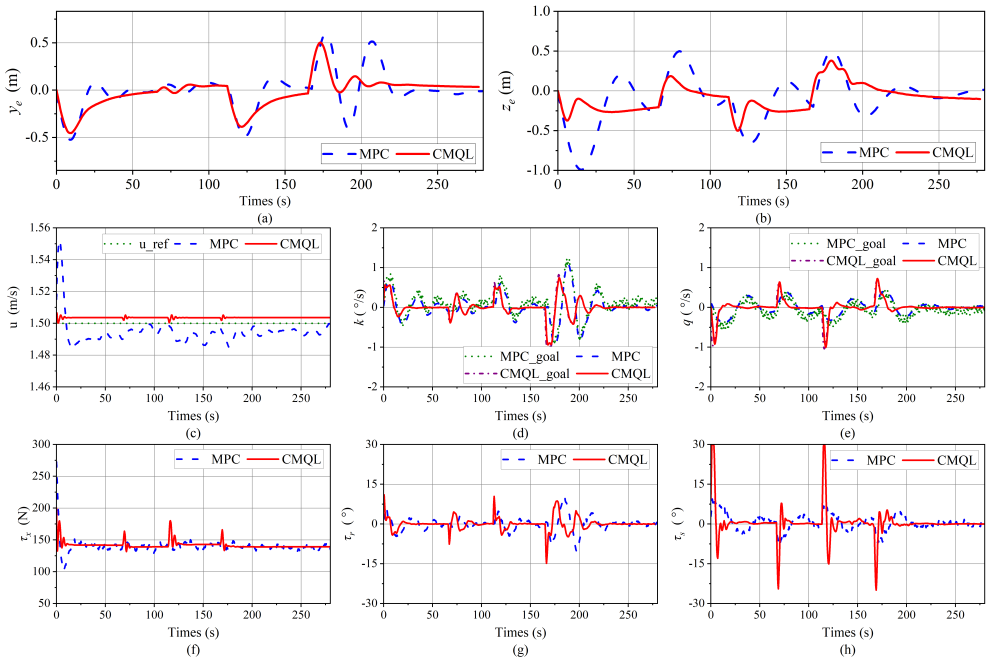

Figure 20. Simulation results of the polyline path-following mission without disturbance. (**a**,**b**) Path-following errors. (**c**) Tracking results of the surge velocity. (**d**,**e**) Tracking results of velocity controllers. The outputs, CMQL_goal and MPC_goal, from the path-following controllers based on CMQL and MPC, are, respectively, employed as inputs to the corresponding velocity controllers. (**f**) Propeller thrust. (**g**) Vertical rudder angle. (**h**) Horizontal rudder angle.

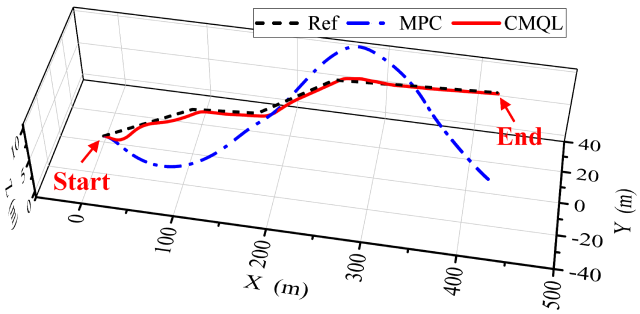

Figure 21. AUV trajectory of the polyline path-following mission under ocean current disturbance.

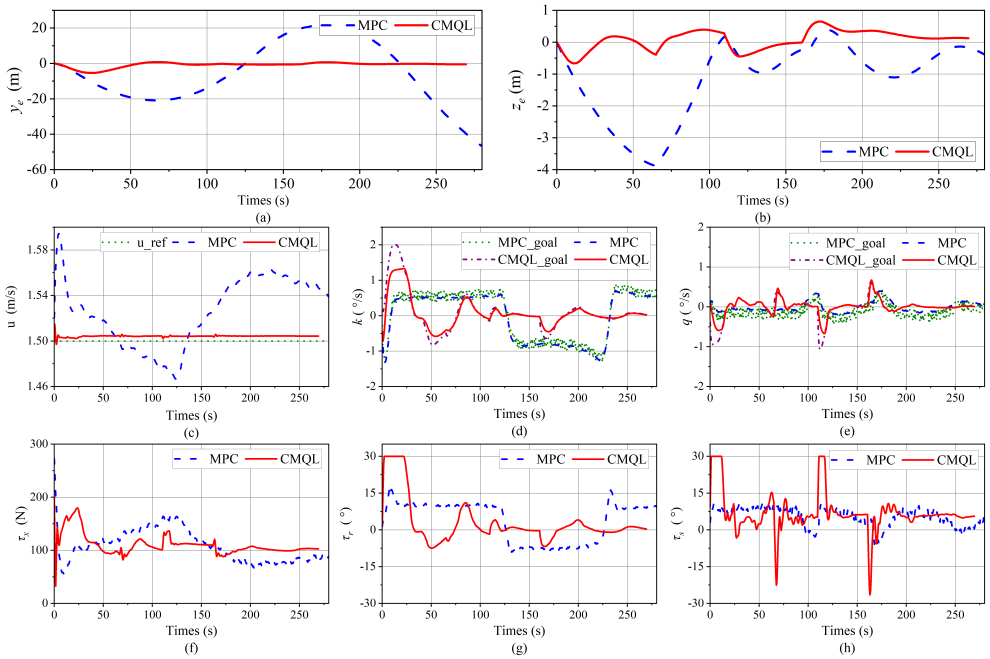

Figure 22. Simulation results of the polyline path-following mission under ocean current disturbance. (**a**,**b**) Path-following errors. (**c**) Tracking results of the surge velocity. (**d**,**e**) Tracking results of velocity controllers. The outputs, CMQL_goal and MPC_goal, from the path-following controllers based on CMQL and MPC, are, respectively, employed as inputs to the corresponding velocity controllers. (**f**) Propeller thrust. (**g**) Vertical rudder angle. (**h**) Horizontal rudder angle.

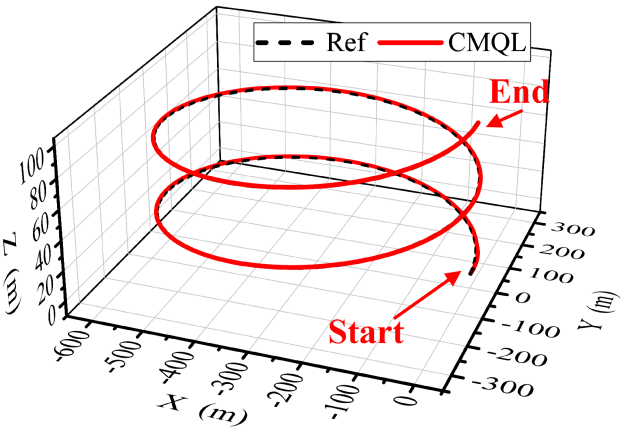

Figure 23. AUV trajectory of the spiral path-following mission under variable ocean current disturbance.

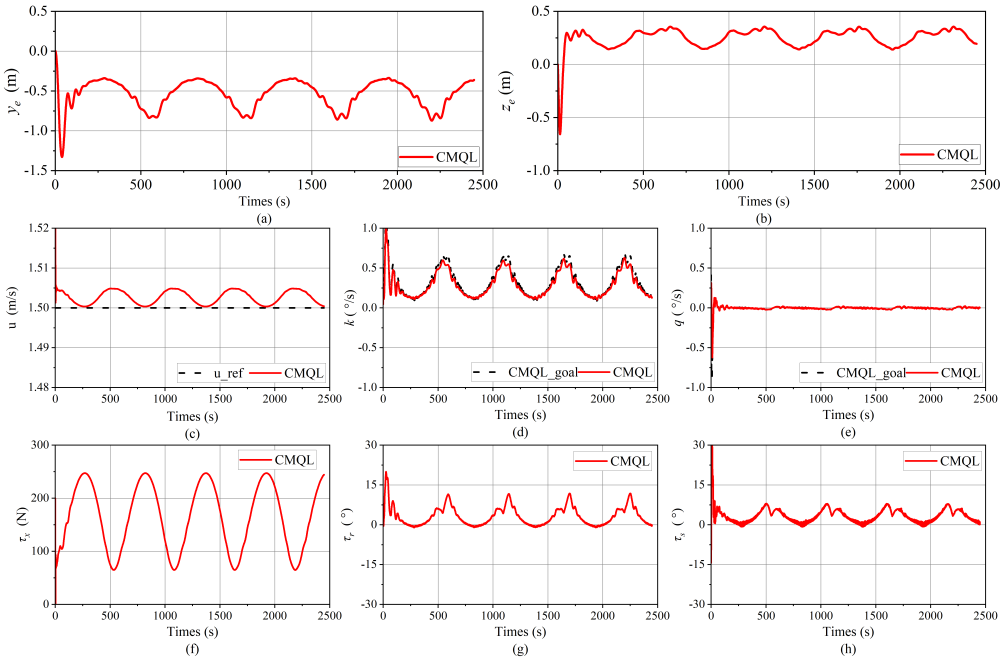

Figure 24. Simulation results of the spiral path-following mission under variable ocean current disturbance. (**a**,**b**) Path-following errors. (**c**) Tracking results of the surge velocity. (**d**,**e**) Tracking results of velocity controllers. The outputs, CMQL_goal, from the path-following controller based on CMQL, are employed as inputs to the corresponding velocity controllers. (**f**) Propeller thrust. (**g**) Vertical rudder angle. (**h**) Horizontal rudder angle.

6. Conclusions

This study proposes the CMQL algorithm, utilizing limited data to achieve AUV path-following control under unknown ocean currents. This method introduces a novel AUV modeling approach based on CNP and performs offline conservative policy optimization using CNP-based models. CMQL constructs a CNP-based AUV model using a limited dataset of 1000 samples. This model can accurately predict the AUV's long-term motion states within the dataset's distribution. The controller performs offline conservative policy optimization through an interaction with the CNP-based AUV model, significantly improving training efficiency and safety. Additionally, a two-stage control structure is proposed, which mitigates the impact of ocean currents on the state space, action space, and reward function. This enables the controller to exhibit strong disturbance rejection capability without the need for additional compensation. Furthermore, DR is employed to further enhance the controller's generalization and disturbance rejection. Finally, CMQL is validated using three velocity controllers and the path-following controller. It is confirmed that path-following can be achieved under unknown ocean currents with only 1000 AUV motion data samples. Moreover, the controller trained using the CNP-based AUV models can be applied directly to the simulation environment without fine-tuning, demonstrating its excellent practicality.

Currently, this article confirms the effectiveness of our method through theoretical analysis and simulation. In our future work, we will conduct further validation through field tests. Additionally, we will explore the impact of datasets on the long-term predictive ability of CNP-based AUV models and work on improving the control accuracy of controllers based on suboptimal models.

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

Funding: This work was supported by the National Natural Science Foundation of China under grant No. 42206196 and the Liaoning Revitalization Talents Program under grant No. XLYC1902032.

Data Availability Statement: The dataset used in this study can be made available upon request from the authors.

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

References

1. Zheng, R.; Lu, H.; Yu, C.; Han, X.; Li, M.; Wei, A. Technical research, system design and implementation of docking between AUV and autonomous mobile dock station. *Robot* **2019**, *41*, 713–721.
2. Palomer, A.; Ridao, P.; Ribas, D. Inspection of an underwater structure using point-cloud SLAM with an AUV and a laser scanner. *J. Field Robot.* **2019**, *36*, 1333–1344. [CrossRef]
3. Ahn, J.; Yasukawa, S.; Sonoda, T.; Nishida, Y.; Ishii, K.; Ura, T. An optical image transmission system for deep sea creature sampling missions using autonomous underwater vehicle. *IEEE. J. Oceanic Eng.* **2020**, *45*, 350–361. [CrossRef]
4. Yin, Q.Q.; Shen, Y.; Li, H.J.; Wan, J.H.; Wang, D.R.; Liu, F.X.; Kong, X.R.; He, B.; Yan, T.H. Fuzzy PID motion control based on extended state observer for AUV. In Proceedings of the IEEE Underwater Technology, Kaohsiung, Taiwan, 16–19 April 2019.
5. Huang, Z.J.; Zheng, H.; Wang, S.T.; Liu, Y.J.; Ma, J.; Liu, Y.H. A self-searching optimal ADRC for the pitch angle control of an underwater thermal glider in the vertical plane motion. *Ocean Eng.* **2018**, *159*, 98–111. [CrossRef]
6. Dai, P.L.; Lu, W.J.; Le, K.; Liu, D.K. Sliding mode impedance control for contact intervention of an I-AUV: Simulation and experimental validation. *Ocean. Eng.* **2020**, *196*, 106855. [CrossRef]
7. Zhou, G.Z.; Xiang, X.B.; Liu, C. Parameter identification and model prediction path following control of underactuated AUV: Methodology and experimental verification. *Control. Eng. Pract.* **2023**, *141*, 105729. [CrossRef]
8. Wang, Y.X.; Hou, Y.C.; Lai, Z.N.; Cao, L.L.; Hong, W.R.; Wu, D.Z. An adaptive PID controller for path following of autonomous underwater vehicle based on soft actor-critic. *Ocean. Eng.* **2024**, *307*, 118171. [CrossRef]
9. Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. Soft actor-critic algorithms and applications. *arXiv* **2018**, arXiv:1812.05905.
10. Ma, D.F.; Chen, X.; Ma, W.H.; Zheng, H.R.; Qu, F.Z. Neural network model-based reinforcement learning control for AUV 3-D path following. *IEEE Trans. Intell. Veh.* **2024**, *9*, 893–904. [CrossRef]
11. Fan, Y.X.; Dong, H.Y.; Zhao, X.W.; Denissenko, P. Path-following control of unmanned underwater vehicle based on an improved TD3 deep reinforcement learning. *IEEE Trans. Contr. Syst. Tech.* **2024**, *32*, 1904–1919. [CrossRef]
12. Wang, L.; Zhang, W.; He, X.F.; Zha, H.Y. Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. In Proceedings of the 24th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, London, UK, 19–23 August 2018; pp. 2447–2456.
13. Yu, F.; Chen, H.F.; Wang, X.; Xian, W.Q.; Chen, Y.Y.; Liu, F.C.; Madhavan, V.; Darrell, T. BDD100K: A diverse driving dataset for heterogeneous multitask learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 2633–2642.
14. Kalashnikov, D.; Irpan, A.; Pastor, P.; Ibarz, J.; Herzog, A.; Jang, E.; Quillen, D.; Holly, E.; Kalakrishnan, M.; Vanhoucke, V.; et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In Proceedings of the 2nd Conference on Robot Learning, Zurich, Switzerland, 29–31 October 2018; pp. 651–673.
15. Ma, C.Z.; Yang, D.Y.; Wu, T.Y.; Liu, Z.Y.; Yang, H.X.; Chen, X.Y.; Lan, X.G.; Zheng, N.N. Improving offline reinforcement learning with In-sample advantage regularization for robot manipulation. *IEEE. T. Neur. Net. Lear.* **2024**, 1–13. [CrossRef] [PubMed]
16. Guan, J.Y.; Gu, S.D.; Li, Z.J.; Hou, J.; Yang, Y.Q.; Chen, G.; Jiang, C.J. UAC: Offline reinforcement learning with uncertain action constraint. *IEEE T. Cogn. Dev. Syst.* **2024**, *16*, 671–680. [CrossRef]
17. Rafailov, R.; Yu, T.H.; Rajeswaran, A.; Finn, C. Offline reinforcement learning from images with latent space models. In Proceedings of the 3rd Annual Conference on Learning for Dynamics and Control (L4DC), ETH Zurich, Zurich, Switzerland, 7–8 June 2021; pp. 1154–1168.
18. Kumar, A.; Fu, J.; Tucker, G.; Levine, S. Stabilizing off-policy q-Learning via bootstrapping error reduction. In Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS), Vancouver, BC, Canada, 8–14 December 2019; pp. 11761–11771.

19. Wu, Y.; Tucker, G.; Nachum, O. Behavior regularized offline reinforcement learning. *arXiv* **2019**, arXiv:1911.11361.
20. Kumar, A.; Zhou, A.; Tucker, G.; Levine, S. Conservative q-Learning for offline reinforcement learning. *arXiv* **2020**, arXiv:2006.04779.
21. Kidambi, R.; Rajeswaran, A.; Netrapalli, P.; Joachims, T. MOReL: Model-based offline reinforcement learning. *arXiv* **2020**, arXiv:2005.05951.
22. Yu, T.; Thomas, G.; Yu, L.; Ermon, S.; Zou, J.; Levine, S.; Finn, C.; Ma, T. MOPO: Model-based offline policy optimization. *arXiv* **2020**, arXiv:2005.13239.
23. Yu, T.; Kumar, A.; Rafailov, R.; Rajeswaran, A.; Levine, S.; Finn, C. COMBO: Conservative offline model-based policy optimization. *arXiv* **2021**, arXiv:2102.08363.
24. Deisenroth, M.; Rasmussen, C.E. PILCO: A model-based and data-efficient approach to policy search. In Proceedings of the 28th International Conference on International Conference on Machine Learning, Bellevue, WA, USA, 28 June 2011; pp. 465–472.
25. Kumar, V.; Todorov, E.; Levine, S. Optimal control with learned local models: Application to dexterous manipulation. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 378–383.
26. Kaiser, L.; Babaeizadeh, M.; Milos, P.; Osinski, B.; Campbell, R.H.; Czechowski, K.; Erhan, D.; Finn, C.; Kozakowski, P.; Levine, S.; et al. Model-based reinforcement learning for atari. *arXiv* **2019**, arXiv:1903.00374.
27. Garnelo, M.; Rosenbaum, D.; Maddison, C.J.; Ramalho, T.; Saxton, D.; Shanahan, M.; Whye Teh, Y.; Rezende, D.J.; Eslami, S.M.A. Conditional Neural Processes. *arXiv* **2018**, arXiv:1807.01613.
28. Williams, G.; Aldrich, A.; Theodorou, E.A. Model predictive path integral control: From theory to parallel computation. *J. Guid. Control Dynam.* **2017**, *40*, 344–357. [CrossRef]