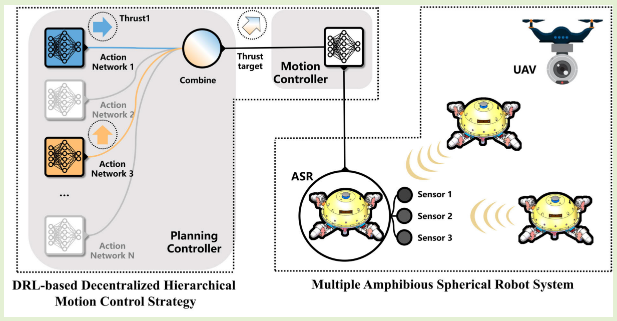
A Deep Reinforcement Learning-Based Decentralized Hierarchical Motion Control Strategy for Multiple Amphibious Spherical Robot Systems With Tilting Thrusters

He Yin, *Member, IEEE*, Shuxiang Guo, *Fellow, IEEE*, Ao Li, *Student Member, IEEE*, Liwei Shi, *Member, IEEE*, and Meng Liu, *Member, IEEE*

***Abstract*—The variable operating conditions and hostile environments faced by underwater robots remain a challenge for motion control in unknown environments. In order to improve the capability of the amphibious spherical robot (ASR) in unknown environments, a decentralized hierarchical deep reinforcement learning (DRL) motion control method based on deep deterministic policy gradient (DDPG) for multiple ASRs system is proposed. In the low-level, a DDPG-based motion controller is trained under a compound rewarding to learn to set the configurations of the tilting angle and rotational speed of each thruster at a proper timescale. At the high-level, a planning controller consisting of different action networks is designed to generate a reasonable thrust target to guide the movement of the robot. Specifically, inspired by the artificial potential field (APF) method, the complex underwater motion can be decomposed into several simple virtual forces. Each action network is trained to learn to generate a virtual thrust target component for a specific action. By combining the outputs of several action networks, the distributed cooperative motion control for multirobot systems can then be easily achieved. Finally, the motion control strategy is applied to the physical multi-ASR system, and the experiment results have shown satisfactory performance.**



***Index Terms*—Amphibious spherical robot (ASR), collision avoidance, deep reinforcement learning (DRL), motion control, multirobot system, thrust-vectoring, tilting thruster.**

I. Introduction

In recent years, underwater robots have gradually attracted people's attention as the main tool for ocean exploration. With the advantages of low cost and excellent mobility [1], [2], [3]. Various small-scale underwater robots have been developed to accomplish many difficult tasks, such as submarine pipelines, submarine cables, and ocean power plants. For underwater robots, complex tasks often require high functionality and precise motion control. It is obvious that the cooperative strategy of multiple robots is an essential tool for solving the above problem. Under the control of the cooperative strategy, the multirobot system can solve complex tasks that cannot be accomplished by a single robot even with expensive equipment. In addition, precise motion control should also be taken into consideration to guarantee the efficiency of task completion. Achieve energy efficient motion control and decentralized multiple robot control with high generalization ability remains challenging.

Over the years, there have been intensive efforts toward the development of autonomous control strategies for underwater robot with fixed thrusters. Underwater robots with fixed thrusters are underwater platforms employed in various engineering and industrial applications. Such popularity results from the simplicity of the mechanical structure and the ease of position and orientation control. Most underwater robots rely on a multifixed thruster mechanism to achieve underwater multidegree-of-freedom (DOF) control. This causes the robot's size, weight, and power consumption to increase.

Manuscript received 23 April 2023; revised 2 October 2023; accepted 10 November 2023. Date of publication 23 November 2023; date of current version 2 January 2024. The associate editor coordinating the review of this article and approving it for publication was Prof. Mehdi Javanmard. *(Corresponding author: Shuxiang Guo.)*

He Yin, Ao Li, Liwei Shi, and Meng Liu are with the Beijing Institute of Technology, Beijing 100081, China (e-mail: yinhe@bit.edu.cn).

Shuxiang Guo is with the Department of Electronic and Electrical Engineering, Southern University of Science and Technology, Shenzhen, Guangdong 518055, China, and also with the School of Life Science and the Key Laboratory of Convergence Medical Engineering System and Healthcare Technology, Ministry of Industry and Information Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: guo.shuxiang@sustech.edu.cn).

Digital Object Identifier 10.1109/JSEN.2023.3333872

Lakhekar et al. [4] combined the disturbance observer and fuzzy S-surface technology in the adaptive control scheme to realize the trajectory tracking of the full-drive autonomous underwater vehicle (AUV). Based on the nonlinear model predictive control (MPC), Heshmati-Alamdari et al. [5] proposed a robust trajectory tracking control for underactuated AUVs. Obstacles, workspace boundaries, and bounds of the vehicle velocity are considered during the control design. Considering the disturbance caused by the manipulator and unknown external disturbances, an observer-based control scheme for an underwater vehicle–manipulator system is developed. This method integrates an adaptive tracking differentiator, an extended state observation, and a fuzzy-based controller, which effectively deals with the problem of complex manipulation in practical scenarios [6], [7]. Based on previous work, they also designed a collision-free control framework in dynamic environments using fuzzy artificial potential field (APF) and sliding mode controllers [8]. By adaptively adjusting the attraction function of APF, biomimetic underwater vehicles can realize obstacle avoidance planning. Compared with traditional fixed thrusters, the variable tilting angle of thrusters and the smaller thrusters improve the maneuverability and load capacity of robots. But the model of the robot with a tilting thruster is nonlinear, which makes the design of the controller more complicated. Bak et al. [9] designed a six-DOF underwater vehicle with four thrusters. The four thrusters around the robot can be rotated independently in the vertex positions. In order to generate the desired thrust, a controller using the decomposition and compensation method based on the actuation model is proposed in this article to configure the tilting angle and the rotational speed of the thrusters. Li and Guo [10] proposed an adaptive multimode switching strategy for the spherical underwater robot. The spherical underwater robot is equipped with four water jets and two propellers. The robot can select the appropriate power equipment according to the current state to achieve precise position control. In recent years, the development of reinforcement learning (RL) technology provides a new solution for the intelligent control of the AUV. Knudsen et al. [11] designed a dual controller for a six DOFs underwater vehicle. The dual controller design includes a deep deterministic policy gradient (DDPG) algorithm for the horizontal movement in conjunction with a PD controller for the vertical movement. Zhang et al. [12] proposed a deep reinforcement learning (DRL) based approach for path-following control of a fish-like robot and the performance of the controller is verified by real-world experiments. In [13], a DRL-based approach is proposed for robotic fish to realize attitude holding control. The method successfully realizes the sim-to-real transfer and directly deployed on a physical robotic fish to hold its attitude in the real world.

Much effort has been devoted to the cooperative mechanism and strategy for multirobot. The cooperation of several robots can perform tasks that a single robot cannot complete. In that sense, the shortcomings of a single robot with low payload, limited sensing capability, and poor battery life can be ignored by the cooperation of multiple robots. Liu et al. [14] proposed a multirobot formation control strategy suitable for constrained space. By introducing the role switching triggered and the formation scaling factor, the size of the formation can change autonomously according to the environment state with limited communication. An et al. [15] proposed an improved leader-follower formation control strategy for multiple spherical underwater robots. Each robot is a leader or follower of the formation and the formation can continue to work despite the failure of a specific robot. With the breakthrough of machine learning, the development of RL provides an efficient solution to multirobot cooperative control. To resolve the USV formation path-following problem, Zhao et al. [16] proposed a formation and path-following control via DRL. The controller trained by the proposed novel random brake mechanism has an excellent performance in formation keeping. Using the RL-based methods, a decentralized circle formation control for fish-like robots is achieved in the simulation and real-world experiment by Zhang et al. [17]. However, most of the RL-based control research for the cooperation of multirobot is limited to computer simulation. In addition, the multiagent RL methods they used focused on the path planning of multirobots, rarely considering the actual physical state of the robots.

In this work, a decentralized hierarchical motion control strategy for multiple amphibious spherical robot (ASR) systems using hierarchical DRL is proposed. By using RL, the low-level controller (motion controller) operating at a fine timescale provides a thrust-vectoring mechanism for ASR based on multiple servos and thrusters. At a larger timescale, the high-level controller (planning controller) guides the generation of thrust from the motion controller according to the state of self and environment. Inspired by the APF method, a complex action in the planning controller is treated as a combination of several simple action components, such as path following, obstacle avoidance, and cooperative escort. Similar to force synthesis, complex motion control of multirobot systems can be easily achieved by combining the outputs of several action networks. The use of RL makes the design of AUV controllers with tilting thrusters easier, while allowing skills to be defined via objective functions. The modular design of the controller makes the design of the reward function independent. It also reduces the training time of each modular network and enables each modular network to be replaced to perform different tasks. The main contribution of this study, relative to other works, is as follows.

1) A DRL-based thrust-vectoring mechanism (motion controller) for the ASR with the tilting thruster is proposed. A compound reward is designed to solve multiobjective optimization problem of the thruster configuration.
2) Inspired by the APF method, a virtual force-based planning controller for multirobot system is designed. The complex underwater robot motion can be described as a combination of several simple virtual forces in the proposed method. This modular design based on virtual forces makes RL training easier and simplifies the design of the reward function.
3) Our approach successfully applies RL-based method to the physical multi-ASR system for motion control, and the experiment results have shown satisfactory performance.

The rest of this article is organized as follows. Section II presents a brief introduction about mutli-robot system. Section III presents the general framework of our proposed method for the ASRs. Simulation and results are provided in Section IV in order to assess the performance of the proposed approach. The effectiveness and robustness of the proposed motion control method are verified through experiments with custom-built robots in Section V. Section VI provides a conclusion of the whole article.

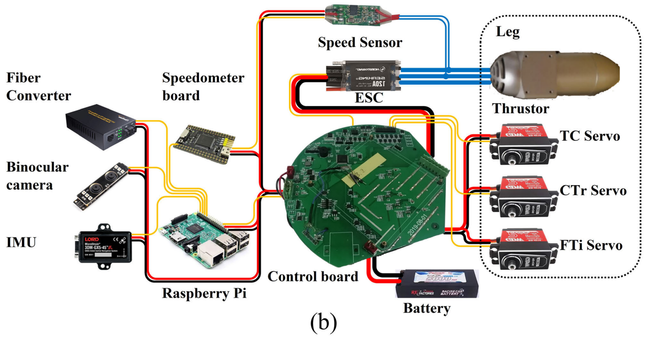
II. System Overview of the Multiple ASRs

A. Structure of ASR

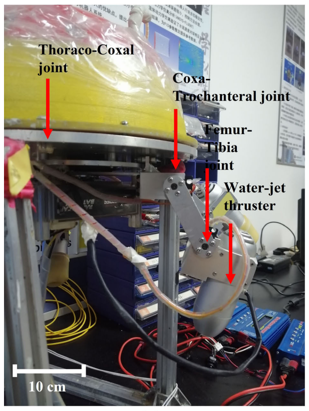
As an upgraded version of the robotic mentioned in the references [18], [19], [20], its mechanical structure is inspired by the morphology and locomotion modes of a turtle. The overall mechanical design of the ASR is shown in Fig. 1(a), which is mainly composed of a sealed hemispherical cabin and four composite mechanical legs evenly distributed on the fuselage. The diameters of the body and the total mass are 0.6 m and 6.7 kg, respectively. Components of the prototype and their wiring are described in Fig. 1(b). Sensors and control modules are located in the hemispherical cabin, which are mainly used to obtain state information of itself and the environment and to control the robot. The sensors equipped on the robot mainly include binocular cameras, inertial measurement unit (IMU) and pressure sensors. A raspberry Pi control board is used to process status information and publish control commands. The four mechanical legs located in the lower half of the robot are used to help the robot realize the movement on land and underwater. After extensive tests, the maximum speed of this amphibious robot in the water can reach 60 cm/s, and the maximum crawling speed of the robot on land is 6.05 cm/s.

In order to realize the precise motion control of the ASR, inspired by the crawling motion of the turtle and the jet propulsion method of the squid, a composite robotic leg is designed that integrates land crawling and underwater vector propulsion. The structure of the leg is shown in Fig. 1(c) and (d). To help the robot crawl on land, each mechanical leg consists of three joints and three links, respectively. The three joints are the thoraco-coxal joint (TC joint), the coxa-trochanteral joint (CTr joint), and the femur-tibia joint (FTi joint). The three links are the “hip” link, the “femoral” link, and the “tibial” link. The angle of each joint is controlled by three servos: TC servo, CTr servo, and FTi servo. The angle range of each joint is 90°. To help the robot swim in the water, the end of each leg is equipped with a thruster. As shown in Fig. 1(b), four electrical speed controller (ESC) controls the speed of the four thrusters. Four external integrated speed sensors capture the real-time speed of the four thrusters and transmit the thruster’s real-time status to the speedometer board.

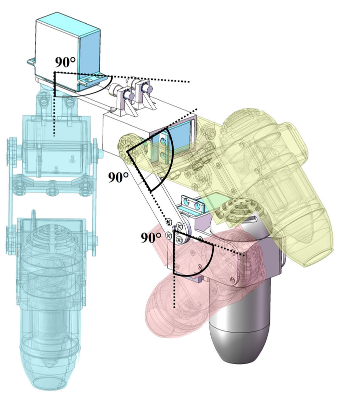
(a)



(b)



(c)



(d)

Fig. 1. Structure of the ASR. (a) Robotic prototype of the multi-ASR system. (b) Schematic of the ASR. (c) Leg structure of the ASR. (d) Range of motion for each joint.

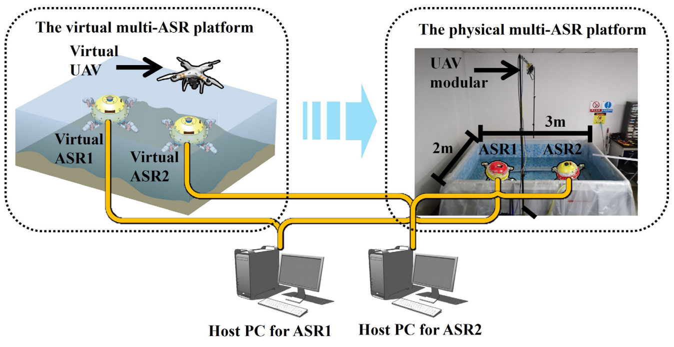


Fig. 2. Structure of the multi-ASR system.

B. Principle of Multi-ASR System

As shown in Fig. 2, the multi-robot experiment platform consists of two parts: the physical multi-ASR platform and the virtual multi-ASR platform. The physical experiment platform includes a 3 $\times$ 2 m pool, a unmanned aerial vehicle (UAV) module, and two ASRs. Through the multirobot images captured by the camera equipped on the UAV module, the UAV module can calculate the position information of each robot and publish this information at a certain frequency. In order to accelerate the training of RL, the main control part of the algorithm is deployed on the corresponding host PC. The robot configures the servos and thrusters based on control commands from the host PC. In the experimental

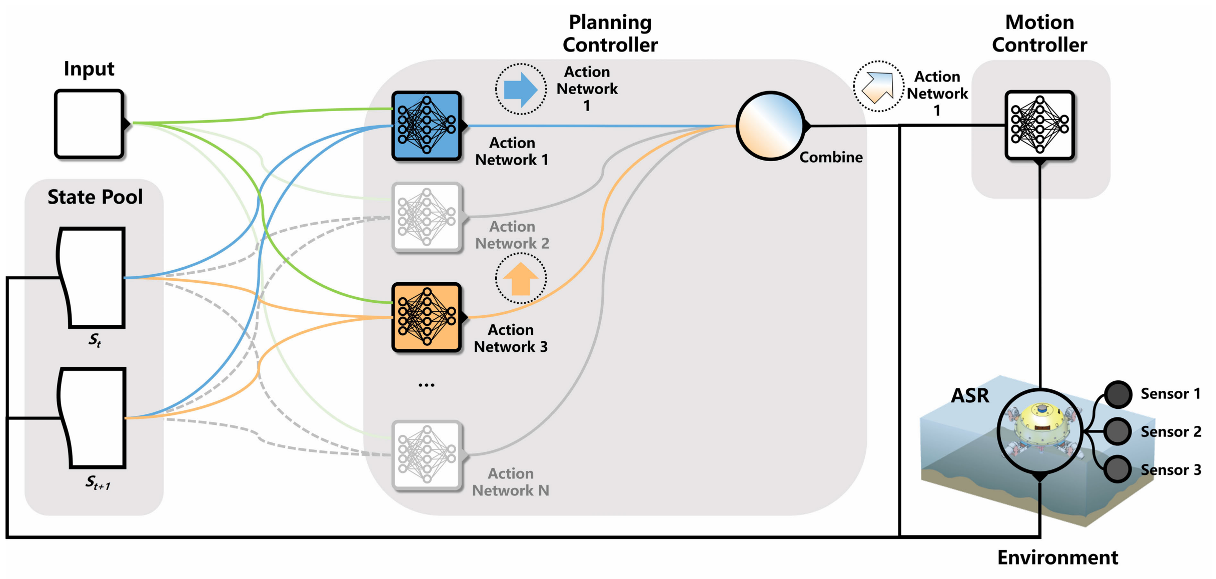


Fig. 3. General framework for the proposed hierarchical RL neural network.

platform, each control system independently perceives the external environment information and outputs the decision to control itself. The communication between each robot relies on a WIFI module. In order to facilitate the transfer of RL algorithms, the configuration of the virtual environment is similar to that of the physical robot multirobot platform. The structure of the virtual simulation platform robot is the same as that of the physical robot, which is imported from the Solidwork model of the robot. The position information of the virtual platform robot is provided by a virtual UAV. The difference from the physical platform is that the size of the virtual pool is 6 $\times$ 6 m. In addition, the virtual robot and the real robot share the same remote host PC. During the training process, the control strategy based on RL is trained on the virtual simulation platform, and then it is directly applied to the actual robot platform for verification.

III. Hierarchical Deep RL Motion Control for Multi-ASR System

To fulfill the motion control of ASR, a hierarchical DRL motion control method based on DDPG is proposed. First, the overall structure of the proposed method is given, and then, the workflow of our method is introduced. Second, the specific design of state representation, action space, and reward functions are introduced. Then, the training objectives and training process for DRL are summarized.

A. DRL Structure for ASR Motion Control

Fig. 3 presents the general framework of the proposed motion control method based on the hierarchical DRL for the multi-ASR system. The proposed motion control method consists of the following two-level controllers: the motion controller (low-level) and the planning controller (high-level). The planning controller operates at the timescale of 2 Hz, while the motion controller operates at 20 Hz. Based on the thrust target from the planning controller and self-state from sensors, the motion controller can directly operate the servos and thrusters of the robot to generate robust thrust that satisfies the thrust target. By invoking motion requirements and environment state obtained from sensors, the planning controller generates thrust targets to guide the motion controller to achieve the desired movement. In particular, inspired by APFs, the complex motion in the planning controller is treated as a combination of several simple actions. By combining the thrust-targets output by each action network in the planning controller, the robot can achieve complex motion control. The state of robots and environment is stored in the state pool. Each action network in the planning controller can select the corresponding information from the state pool as needed.

The training of the proposed motion control method is a process of interacting with the environment iteratively. The motion controller and planning controller are trained separately. In the training process, the motion controller based on the self-state and thrust target from the planning controller generates a control command of robot's servos and thrusters for the desired motion control. Then, the motion controller gets the evaluation from the reward function and adjusts the current action to learn the ideal behavior policy. The above process is repeated until the motion controller can generate the desired thrust. The planning controller will activate the corresponding action network according to the motion requirements. During the training process of each action network, it selects the required information from the state pool and generates a reasonable thrust target. The performance of each action network is evaluated by the action-specific reward function. In this way, the action network can learn the specified basic behavior policy. By combining thrust targets from the action network, complex motion control can be achieved.

B. Motion Controller (Low-Level)

To achieve precise regulation of robot position and orientation while subjected to disturbances, many control methods have been extensively studied with existing overactuated

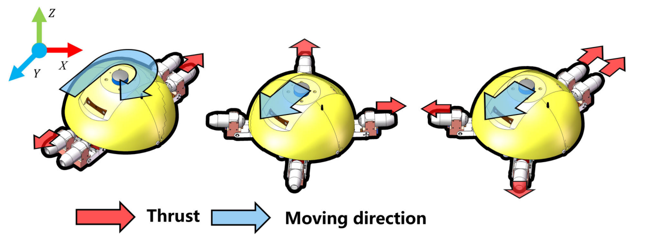


Fig. 4. Relationship between the torque and the configuration of four legs.

underwater robots. In our previous research, the configuration of four legs is set to the X-shape or H-shape for sailing in the water, as shown in Fig. 4. However, such motion patterns do not use the full set of tilting thrusters. In the proposed motion control method, a DDPG-based motion controller was designed to coordinate steering gears and thrusters for generating stable thrust which can track the thrust target from the planning controller. At each time step, the motion controller will generate a reasonable action ${}^{M}\boldsymbol{A}$ based on the observation of its own state ${}^{M}\boldsymbol{S}$ and the thrust-target ${}^{P}\boldsymbol{F}$ from the planning controller. Then the performance of the motion controller is evaluated by the reward function and gets a reward ${}^{M}R$. This process is repeated until the motion controller can generate the required thrust.

The state space ${}^{M}\boldsymbol{S}$ of the motion controller represents the local environmental information of ASR at the current moment. It includes local state variable ${}^{M}\boldsymbol{S}_1$ and energy consumption variable ${}^{M}\boldsymbol{S}_2$. These state variables can ensure that the robot can generate the desired thrust in a low energy consumption manner. ${}^{M}\boldsymbol{S}_1$ includes the joint angles and the rotational speed of the thruster. Since only the motion of the robot in the 2-D plane is considered, the CTr and FTi joints of the four legs of the robot are fixed. Then ${}^{M}\boldsymbol{S}_1$ can be represented as ${}^{M}\boldsymbol{S}_1 = [{}^{M}\boldsymbol{S}_a, {}^{M}\boldsymbol{S}_t] = [{}^{M}S_{a1}, {}^{M}S_{a2}, {}^{M}S_{a3}, {}^{M}S_{a4}, {}^{M}S_{f1}, {}^{M}S_{f2}, {}^{M}S_{f3}, {}^{M}S_{f4}]$, where ${}^{M}\boldsymbol{S}_a$ is the angle of the TC joints of the four legs, and ${}^{M}\boldsymbol{S}_t$ is the speed of the four thrusters. These two state variables impact the actual thrust generated by the robot. The energy consumption variable is defined as ${}^{M}\boldsymbol{S}_2 = [S^{M}_{\text{ea}}, S^{M}_{\text{ef}}]$. S^{M}_{ea} represents the total energy consumed by four TC servos, which can be expressed as ${}^{M}S_{\text{ea}} = \sum_{l=1}^{4} |{}^{M}S_{\text{al}}(t) - {}^{M}S_{\text{al}}(t-1)|$. ${}^{M}S_{\text{al}}(t-1)$ represents the status of TC joint servo in the last step. ${}^{M}S_{\text{ef}}$ represents the total energy consumed by the four thrusters. The value can be obtained as ${}^{M}S_{\text{ef}} = \sum_{l=1}^{4} {}^{M}S_{\text{fl}}(t)$. Combined, the state features create a 10-D state space. Therefore, the state space of the motion controller can be described as ${}^{M}\boldsymbol{S} = [{}^{M}\boldsymbol{S}_1, {}^{M}\boldsymbol{S}_2]$.

The action space of the motion controller is a set of actions that can be selected at each step. As shown in Fig. 5, the ASR is assumed to navigate only in 2-D space. By configuring servos and thrusters, the motion controller can generate the desired thrust to achieve precise position control. Due to the physical constraints of the ASR, the angle of the steering gear is limited as Angel $\in [-45^\circ, 45^\circ]$ and the speed of the thruster is limited as Thrust $\in [0, 300]$. Due to the operating frequency of the motion controller, the angle action and thrust action are limited to ${}^{M}A_a \in [-1^\circ, 1^\circ]$ and ${}^{M}A_t \in [-1, 1]$. Therefore, the action space of the control layer can be described as ${}^{M}\boldsymbol{A} = [{}^{M}A_{a1}, {}^{M}A_{a2}, {}^{M}A_{a3}, {}^{M}A_{a4}, {}^{M}A_{f1}, {}^{M}A_{f2}, {}^{M}A_{f3}, {}^{M}A_{f4}]$.

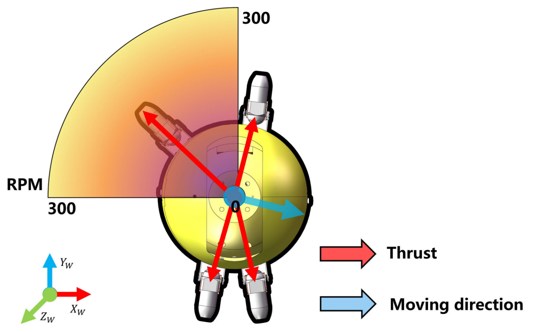


Fig. 5. Action space of motion controller.

The design of the reward function should satisfy the following requirements: 1) minimize the error between actual thrust and thrust target and 2) reduce the energy consumption of four legs. To achieve the above goals, two reward functions are designed independently and combined together to address the merged motion control problem.

The desired thrust output from the motion controller is the guarantee of precise movement. Therefore, the reward function based on the error between the actual thrust and the thrust target is designed to evaluate the quality of the generated thrust. Generally, ASR can achieve the desired motion control when the error is minimal. Thus, the Euclidean distance between the actual thrust and the thrust target is taken as the input of the reward function. The ${}^{M}R_{\text{error}}$ can be defined as

$$ {}^{M}R_{\text{error}} = -\left\| {}^{P}F - T\left({}^{M}S_a, {}^{M}S_t\right) \right\| \tag{1} $$

where ${}^{P}F$ is the thrust target from the planning controller, and T is a mapping that translates the robot state into the actual thrust. During the underwater movement of the robot, most of the energy consumption of the robot comes from the energy consumption of the steering gears and thrusters. In order to improve energy efficiency, the energy consumed by the robot should also be considered. Therefore, the energy reward function is defined as follows:

$$ \begin{aligned} {}^{M}R_{\text{energy}} &= -\sum_{l=1}^{4}\left(\omega_a^{M}E_a + \omega_t^{M}E_t\right) \\ &= -\sum_{l=1}^{4}\left(\omega_a\left({}^{M}S_{\text{al}}(t) - {}^{M}S_{\text{al}}(t-1)\right) + \omega_t^{M}S_{\text{fl}}(t)\right) \end{aligned} \tag{2} $$

where ${}^{M}E_a$ is the energy consumption of servos, ${}^{M}E_t$ is the energy consumption of thrusters, and ω is the weight. In the reward function, inefficient actions generated by the motion controller will be penalized. Since joint servos run most of the time under part load, the weight of the joint servos's energy consumption can be small or ignored.

Following that, the reward function can be:

$$ {}^{M}R = \omega_1\ {}^{M}R_{\text{error}} + \omega_2\ {}^{M}R_{\text{energy}} \tag{3} $$

where ω_1 and ω_1 are the weights of factors.

C. Planning Controller (High-Level)

In the aquatic environments, some unknown factors, such as the hydrodynamics of the underwater robot and unknown disturbances, remain a challenge for the design of the robot controller. Although learning-based control algorithms can effectively deal with the above problems, long training time and complex reward function design limit their application in underwater robots. For these reasons, a virtual force-based planning controller is proposed. In the planning controller, each simple motion of the robot can be described by a virtual force. In this way, the complex motion of robots can be treated as a combination of a series of simple virtual forces. For example, the trajectory-tracking motion of an underwater robot in obstacle environment can be decomposed into tracking motion and obstacle avoidance motion. The planning controller consists of a series of action networks. Each action network is responsible for generating a virtual thrust target for a specific action. By combining the outputs of several action networks, complex motion control can also be achieved. During the training process, each action network will be trained independently to learn the corresponding simple motion, which not only simplifies the design of the reward function but also shortens the training time of RL. Furthermore, since the motion of the robot is modularized, when part of the action network needs to be modified, only the corresponding action network instead of the whole network needs to be retrained. In addition, classical control methods can also be integrated into the controller in the form of virtual forces.

The planning controller is mainly responsible for decomposing the desired robot motion into each individual simple motion, and generating the thrust-target to guide the motion controller. Different action networks in the planning controller have different network structures, state spaces, and reward functions. Therefore, a state pool is designed in the proposed motion framework that contains all the state information that needs to be monitored. Each action network selects the appropriate state to form its own state input according to its needs. But the output of all action networks is the same 2-D vector ${}^{P}\boldsymbol{A}$. Finally, by evaluating the performance of the planning controller, it will get a reward ${}^{P}R$.

The path-following task is a common task for underwater robots. During the training of the action network, a random target location is randomly generated around the robot. To urge the ASR to quickly reach the target location, the action network in the planning controller is trained to generate a reasonable thrust target at each step. The input of the action network includes the normalized position error and actual thrust generated by the robot, which are also factors that impact the performance of the action network. The normalized position error ${}^{P}S_1^{\text{pf}}$ represents the Euclidean distance between the present position and the expected position of the robot. The observation of actual thrust ${}^{P}S_2^{\text{pf}}$ is input into the action network in the form of polar coordinates, which includes two parts: amplitude and angle of the thrust. Consequently, the input state of the action network is described by ${}^{P}\boldsymbol{S}^{\text{pf}} = [{}^{P}\boldsymbol{S}_1^{\text{pf}}, {}^{P}\boldsymbol{S}_2^{\text{pf}}]$. The reward function for the path following network can be represented as:

$$\begin{aligned} {}^{P}R^{\text{pf}} &= -\left(\omega_1\ {}^{P}R_1^{\text{pf}} + \omega_2\ {}^{P}R_2^{\text{pf}}\right) \\ &= -\left(\omega_1 \frac{\left(\boldsymbol{p}^d - \boldsymbol{p}\right)^M\boldsymbol{F}}{\left\|\left(\boldsymbol{p}^d - \boldsymbol{p}\right)\right\| \left\|{}^{M}\boldsymbol{F}\right\|} + \omega_2 \tanh\left(\left\|\left(\boldsymbol{p}^d - \boldsymbol{p}\right)\right\|\right)\right) \end{aligned} \tag{4}$$

where $\boldsymbol{p}$ is the present position of the robot; $\boldsymbol{p}^d$ is the expected position; ${}^{M}\boldsymbol{F}$ is the actual thrust. ${}^{P}R_1^{\text{pf}}$ indicates the direction reward function. When the direction of thrust toward the destination, the robot can reach the destination faster and get higher rewards. ${}^{P}R_2^{\text{pf}}$ represents the distance reward function. it implies that the closer the robot is to the destination, the greater the reward it gets.

The objective of obstacle avoidance action is to keep a safe distance between the robot and the obstacle. It requires the action network in the planning controller to learn to generate a target thrust that keeps the robot away from the obstacle while navigating through the obstacle environment. During the training process, an obstacle with a random radius d_r randomly appears within the perception range of the robot with a radius d_o. The state space for the obstacle avoidance task at each time step is defined as ${}^{P}\boldsymbol{S}^{\text{ao}} = [{}^{P}\boldsymbol{S}_1^{\text{ao}}, {}^{P}\boldsymbol{S}_2^{\text{ao}}]$, where ${}^{P}\boldsymbol{S}_1^{\text{ao}}$ is the normalized distance between the nearest obstacle and the robot, ${}^{P}\boldsymbol{S}_2^{\text{ao}}$ is the normalized actual thrust. A feasible solution imitation learning (IL) is used to train the obstacle avoidance action network. The obstacle avoidance action network tries to learn a policy by imitating the expert's behaviors. This training method can make the robot's obstacle avoidance movement more reasonable. In simulation and actual experiments, the expert's behavior is calculated by the APF. The reason for choosing the APF method as expert's behaviors is that it provides simple and effective solutions for practical application. The reference thrust based on APF can be written as

$$ {}^{P}\boldsymbol{F}_d^{\text{ao}} = \begin{cases} 3(-\tanh(2d) + 1.5) \\ \quad \times \dfrac{(\boldsymbol{p} - \boldsymbol{p}_o)}{\left\|(\boldsymbol{p} - \boldsymbol{p}_o)\right\|}, & \left\|(\boldsymbol{p} - \boldsymbol{p}_o)\right\| \le d_o \\ 0, & \left\|(\boldsymbol{p} - \boldsymbol{p}_o)\right\| \ge d_o \end{cases} \tag{5}$$

where $\boldsymbol{p}_o$ is position of the obstacle, $\|(\boldsymbol{p} - \boldsymbol{p}_o)\|$ is the shortest distance between the robot and the obstacle. The design of the obstacle avoidance reward function is also one of the most critical factors, which impact the performance of the obstacle avoidance action. Given the reference thrust ${}^{P}\boldsymbol{F}_d^{\text{ao}}$ and actual thrust ${}^{M}\boldsymbol{F}$, the reward ${}^{P}R^{\text{ao}}$ is designed to encourage the robot to imitate the style of the reference thrust

$$ {}^{P}R^{\text{ao}} = -\left\|{}^{P}\boldsymbol{F}_d^{\text{ao}} - {}^{M}\boldsymbol{F}\right\|. \tag{6}$$

Cooperative escort is an important method to protect the surface ship. Inspired by the movement of underwater fish schools, we design a dynamic circle formation-based escort maneuver for the multi-ASR system. Dynamic circle formation is a common phenomenon observed in fish schools when evading predators. Each robot in the swarm is required to sail around a specific target in a clockwise or counterclockwise circular formation to escort the target. The dynamic circle formation state vector is defined as ${}^{P}\boldsymbol{S}^{\text{es}} = [{}^{P}\boldsymbol{S}_1^{\text{es}}, {}^{P}\boldsymbol{S}_2^{\text{es}}, {}^{P}S_3^{\text{es}}]$. ${}^{P}\boldsymbol{S}_1^{\text{es}}$ is the normalized distance between specific target and the robot. ${}^{P}\boldsymbol{S}_2^{\text{es}}$ is the normalized actual thrust. ${}^{P}S_3^{\text{es}}$ is the error of the escort radius, which is defined as ${}^{P}S_3^{\text{es}} = \text{norm}(\|\boldsymbol{p} - \boldsymbol{p}_{\text{tar}}\| - d_{\text{es}})$. $\boldsymbol{p}_{\text{tar}}$ is the position of the specific target and d_{es} is the error of the escort radius. Similar to the output of the above several action networks, the output of the action network is also a 2-D vector thrust target, which is a component of the thrust target sent to the control controllers. To urge ASRs to sail coherently in a clockwise or counterclockwise circular formation, the actual thrust of the ASR is embodied in the design of our reward function. In the reward function, when the direction of the thrust target is tangent to the circle of the formation, the reward of ASR should be maximized

$$ {}^{P}R_{a}^{\mathrm{es}} = -\frac{\left|\left(\boldsymbol{p}-\boldsymbol{p}_{\mathrm{es}}\right)^{M}\boldsymbol{F}\right|}{\left\|\left(\boldsymbol{p}-\boldsymbol{p}_{\mathrm{es}}\right)\right\|\left\|\left({}^{M}\boldsymbol{F}\right)\right\|}. \tag{7} $$

Besides, in order to limit the range of motion of the robot around the circumference, the distance between ASR and the center of the circumference should be stressed when it comes to the design of reward function

$$ {}^{P}R_{d}^{\mathrm{es}} = -\mathrm{norm}\left(\left\|\boldsymbol{p}-\boldsymbol{p}_{\mathrm{es}}\right\| - d_{\mathrm{es}}\right). \tag{8} $$

Following that, the reward function for the escort action network can be:

$$ {}^{P}R^{\mathrm{es}} = \omega_1\ {}^{P}R_{a}^{\mathrm{es}} + \omega_2\ {}^{P}R_{d}^{\mathrm{es}}. \tag{9} $$

D. Training Algorithm

The basic idea of the RL is to make the agent learn a strategy π to maximize the expected in the process of interaction with the environment. The cumulative reward U can be expressed as

$$ U_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots \tag{10} $$

where $\gamma \in [0, 1]$ is the reward discount factor. The action value function $Q_\pi(s, a)$ represents the reward expectation of discounted cumulative rewards for an agent performing an action in a certain state s and following the policy π thereafter $Q_\pi(s, a) = E(U_t|S_t = s, A_t = a)$.

The DDPG model structure includes an actor network with a parameter of θ^π and a critic network with a parameter of θ^Q. The loss function of the present critic network is defined as

$$ J\left(\theta^Q\right) = \frac{1}{m}\sum_{j=1}^{m}\omega^j\left(y^j - Q\left(s_t^j, a_t^j, \theta^Q\right)\right) \tag{11} $$

where m is the number of samples, ω^j is the weight of sample j, $Q(S_t^j, A_t^j, \theta^Q)$ is the output of the present critic network. y^j is the target action value calculated by the target value network, which can be defined as

$$ \nabla_J(\theta^\pi) = \frac{1}{m}\sum_{j=1}^{m}\left[\nabla_a Q_{(s_i,a_i,w)}|_{s=s_i,a=\pi_\theta(s)}\nabla_\theta\pi_{\theta(s)}|_{s=s_i}\right]. \tag{12} $$

IV. Simulation and Results

In this section, the RL-based motion control method for the multi-ASR system is verified by simulation. Various simulation results are presented and discussed to verify the effectiveness and robustness of the proposed method.

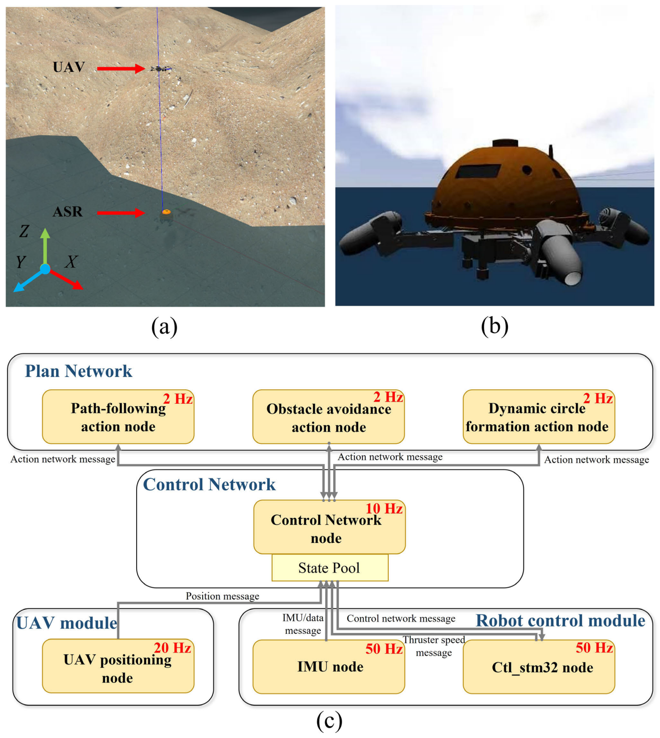


Fig. 6. Simulation environment and program structure. (a) Simulation environment based on the Gazebo. (b) Virtual ASR in simulation environment. (c) Relationship between nodes.

A. Simulation Environments

In order to verify the effectiveness of the proposed motion control method, a simulation environment shown in Fig. 6(a) and (b) with casually obstacles is configured. The simulation environment is constructed based on the extension to the open-source robot simulator Gazebo in underwater scenarios. The ASR simulator uses the robot operating system (ROS) to communicate with the simulation environment. In the simulation environment, the gray cuboids are used as walls that limit the robot's range of motion. Gray spherical objects are used as obstacles.

The proposed RL-based motion control method is implemented with Keras under the ROS. The subscription and publish relationships for messages between nodes is shown in the Fig. 6(c). The planning network is composed of three action network nodes, including path following action node, obstacle avoidance action node, and escorting action node. However, only the selected action network is active. The frequency of different action networks can be different. This can improve the sensitivity of certain actions. Considering the physical limitations of the robot, the frequency of the action network is set at 2 Hz in both simulation and real experiments. The 2-Hz control frequency not only ensures that the robot can effectively perceive environmental information and complete basic underwater operation tasks but also gives servos and thrusters enough response time to generate the desired thrust. The operating frequency of the control network node is set at 20 Hz. In addition, the state pool in the motion controller node is used to store state information from other nodes and publish it to the active action network node. In order to speed

TABLE I
DETAILS OF PARAMETERS

Parameter	Motion Controllor	Path-following	Obstacle avoidance	Escorting
RL learning rate	0.0001	0.0001	0.0001	0.0001
Replay memory size	80000	80000	80000	80000
Batch size	30	30	30	30
Max Episode	180000	1000	180000	1000
Discount factor	0.8	0.8	0.8	0.8

up the training of the RL neural network, both the planning controller and the motion controller are deployed on the corresponding host PC. The IMU node and Ctl_stm32 node are located in the robot, which is used to provide the robot course and thruster speed information. In addition, the Ctl_stm32 node will also convert the leg configuration information from the control network into executable commands and send them to the subordinate control board.

The policies of the proposed RL-based motion control method are trained by two steps. The first step is to train the motion controller based on the above reward function. With the trained motion controller, in the second step, we train planning networks to learn different actions. Details of parameters are given in Table I. A computer with Intel XEON E3-1241 (up to 3.5 GHz) processor, NVIDIA Quadro 2000 GPU, and 8 GB RAM is used for training and testing. Each network requires about two days to be trained.

B. Simulation Results

For the motion controller, the policy should learn to coordinate the tilt angle and rotational speed of the robot's four legs to track the thrust target from the planning controller. The values of the input states and output actions of the networks are normalized to range approximately between $[-1, 1]$. During the training, the motion controller is required to track a reference thrust target for a limited time each turn. On the basis of satisfying the reward function, the smaller the error between the actual thrust generated by the motion controller and the thrust target, the greater the reward the motion controller can get. Fig. 7(a) illustrates the motion controller learning curve. It can be found that the curves converge to a robust performance, which means that the training process of control networks tends to be stable. The output from the motion controller is evaluated and compared with a reference thrust target. The reference thrust target is set as a spiral-shaped curve. Results are shown in Fig. 7(e)–(g). The error between the thrust-target and actual thrust generated by the motion controller is shown in Fig. 7(f) and (g), where Fig. 7(f) is the angle error and Fig. 7(g) is the amplitude error. In this case, the motion controller can coordinate the angles and the thrusts of the robot's four legs to generate the desired thrust within a limited time.

The planning controller includes three action networks: path following action network, obstacle avoidance action network, and escort action network. Each action network is trained independently. In addition, all action networks are trained on the same trained motion controller.

For the path-following action, the maximum reward would require the robot instantly moving to the target location.

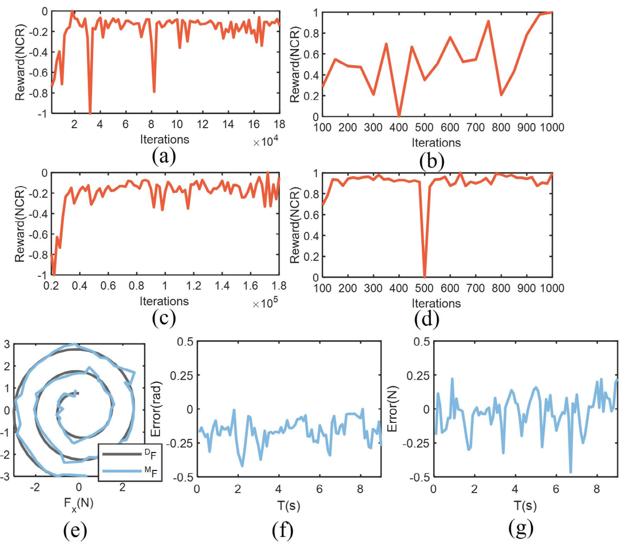


Fig. 7. Training curves of networks and simulation results of motion controller. (a) Curve of motion controller. (b) Curve of path following action network. (c) Curve of obstacle avoidance action network. (d) Curve of escort action network. (e) Comparison of the reference thrust with the actual thrust. (f) Angular error of thrust. (g) Amplitude error of thrust.

Learning curves for path-following action network is available in Fig. 7(b). After 500 training iterations, the generated thrust-target is reasonable and the reward is tremendous. A multirobot trajectory tracking simulation is carried out to evaluate the performance of the action network. In the simulation, robot1 is required to track a 1.5×4 m square trajectory, robot2 follows robot1 at a distance of 2.5 m. The trajectories of the two robots are shown in Fig. 7(a), and the tracking error of the two robots are shown in Fig. 7(b). It can be found that both robot1 and robot2 can follow preset trajectories and maintain the formation shape. The result indicates that the path-following action network has good performance in dealing with the problem of multirobot path-following.

For the obstacle-avoiding action, the action networks should learn to correctly avoid the closest obstacle. The reward penalizes the robot for staying close to the obstacle. Learning curves for obstacle-avoiding action network are available in Fig. 7(c). To better evaluate the obstacle-avoiding action network, we analyze the action outputs from the action networks under the interference of obstacles. The errors between the reference action output from APF and the actual output from the action network in angle and magnitude, respectively, are shown in Fig. 8(c) and (d). When the distance between the obstacle and the robot is greater than 0.05 m, the trained obstacle avoidance action network can generate an output similar to APF. The reason for this phenomenon is that the control network has less training for obstacles with small distances. The results show that the trained obstacle avoidance action network can generate a reasonable thrust target that can keep the robot away from the obstacle. In addition, the performance of the obstacle avoidance action network was also evaluated in the multirobot path-following experiment, the results of which are shown in Fig. 8(a) and (b). In the simulation, a virtual sphere obstacle is considered with centers

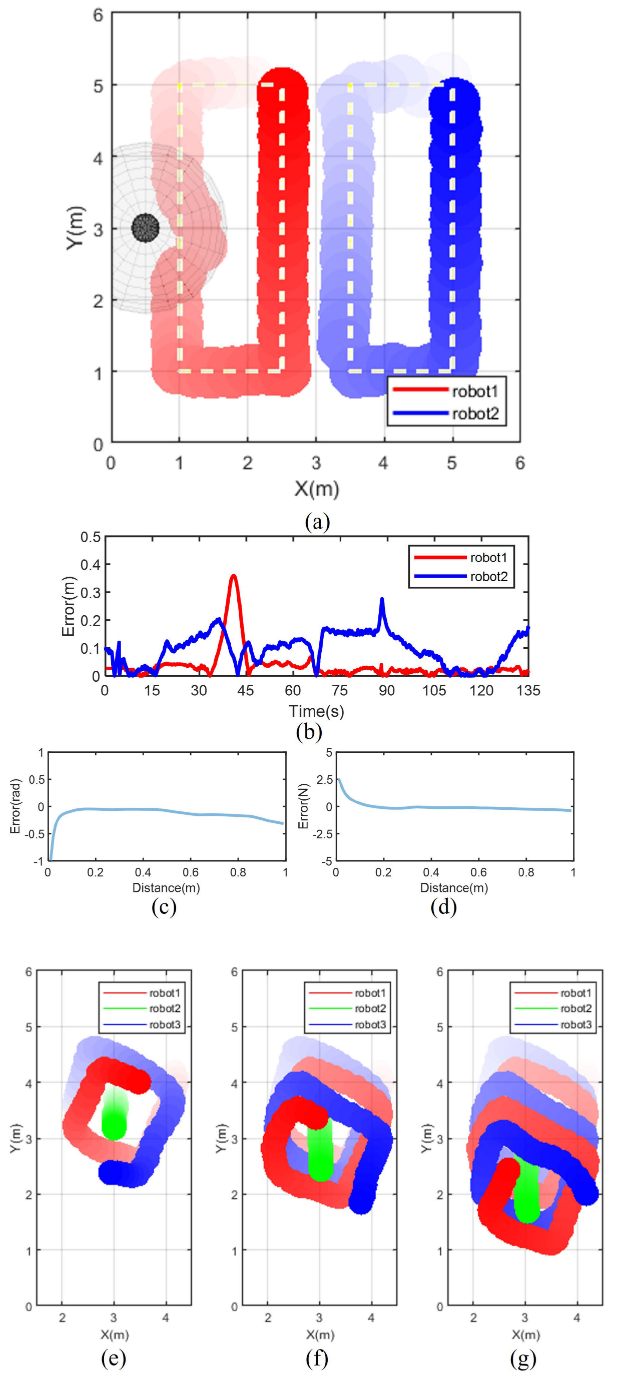


Fig. 8. Simulation results of planning controller. (a) Trajectories of each robot in the simulation. (b) Path following error of two robots. (c) Angular error of thrust. (d) Amplitude error of thrust. (e)–(g) Snapshots of the escort task with three robots.

at $[0, 3]^T$ and a radius of 0.2 m. The safe distance for obstacle avoidance is set to 1.2 m. Robot1 is required to avoid obstacles during path following. The tracking error at 30–40 s is caused by the control of the obstacle avoidance action network. It is observed that the position of the robot is adjusted dynamically and steadily, which ensures the navigation safety of the robot.

The performance of the escort action network is also evaluated by a multirobot composite task. In the simulation, robot2 tracks a straight line trajectory under the control of the path-following action network, while robot1 and robot3 escort robot2 under the cooperative control of the obstacle avoidance action network and the escort action network. A snapshot of the motion trajectories of three robots is presented in Fig. 8(e)–(g). In this simulation, only the escort action network is active for robot2 and robot3 most of the time, which drives the two robots to form the dynamic circle formation around robot1. When the distance between two robots is less than the safe distance, the obstacle avoidance action network and the escort action network will be activated at the same time to allow the robot to make the optimal choice.

V. Experiment on Formation of Multiple ASRs

After the simulations in Section IV, the proposed RL-based motion control method is tested in the real world. Two physical experiments examples are presented to illustrate the effectiveness of the proposed methods. In the physical experiment, the status information of the robot is provided by the UAV module and the sensors equipped on the robot. The speed of thrusters are limited to: Thrust $\in [100, 300]$, the angular of each steering gear is constrained as Angel $\in [-45°, 45°]$. Note that, the lowest nonzero speed of thrusters allows the robot to float in water. Since the input and output of the network are normalized values during the training process of each network, the proposed method trained in simulation can be directly applied on the physical platform. The subscription and publish relationships for messages between nodes in the physical experiment are similar to the simulation experiment, which is shown in Fig. 6(c).

In the first experiment, two robots are required to track a 0.7 $\times$ 1.5 m square trajectory. In addition, robot1 needs to avoid obstacles while tracking the trajectory. The virtual obstacle is considered with centers at $[0, 1.25]^T$ and a radius of 0.2 m. The safe distance for obstacle avoidance is set to 0.8 m. Experiment results are shown in Fig. 9. Fig. 9(a) shows that two robots can successfully track the desired trajectory by locally adjusting the motion trajectory to avoid collisions with obstacles under the cooperative control of path-following and obstacle avoidance action network. It can be seen from Fig. 9(b) that the trajectory error increases when the robot approaches the obstacle and decreases when the robot moves away from the obstacle. In order to show the actual process more clearly, a snapshot of the experiment scenario is presented in Fig. 9(c)–(f). Benefitting from our designed superior motion control strategy, each robot can dynamically change its position to cope with multirobot path-following in obstacle environment.

In the second experiment, the performance of escort task were tested. Due to limitations of the pool size, the protected robot is replaced by a virtual robot whose position is set as $[1.5, 1.25]^T$. Meanwhile, two robots were asked to complete a collaborative escort mission for the virtual robot. The snapshot of the motion trajectories of the two robots is presented in Fig. 10. It can be found that the two robots form a dynamic circle formation centered on the virtual robot under the control of the planning controller. The motion controller generates the desired thrust by configuring the tilt angle and rotational speed of thrusters.

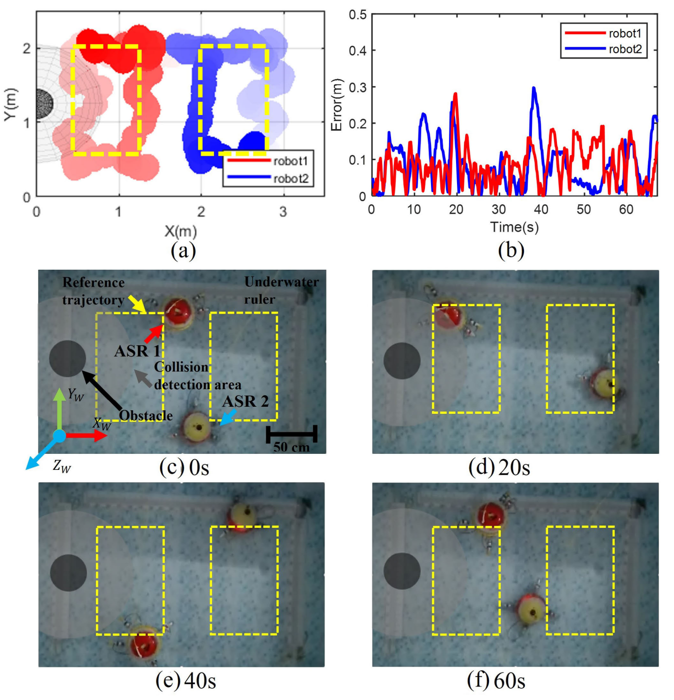


Fig. 9. Results of multirobot trajectory tracking task with one obstacle. (a) Trajectories of two robots. (b) Path-following error. (c)–(f) Snapshots of the multirobot trajectory tracking task.

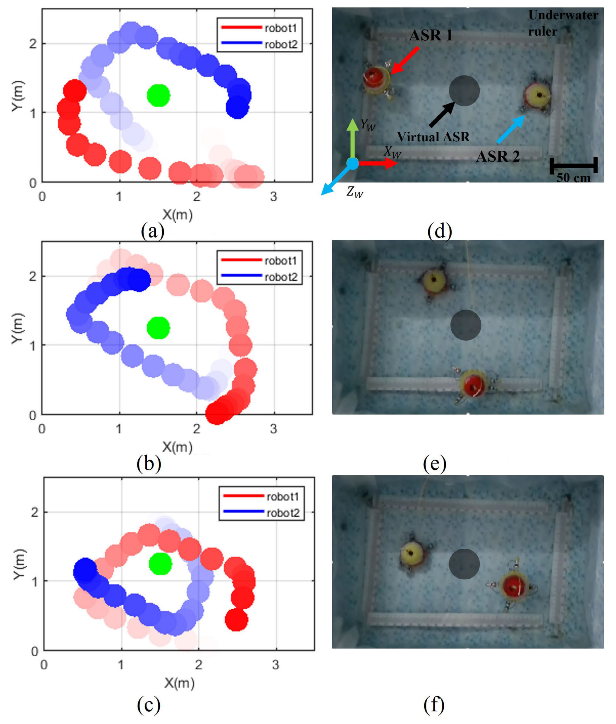


Fig. 10. Snapshots of the multirobot escort task. (a) 0–5 s. (b) 5–10 s. (c) 10–15 s. (d) 0–5 s. (e) 5–10 s. (f) 10–15 s.

VI. Conclusion

First, a DRL-based thrust-vectoring mechanism for multiple ASR systems is designed. By introducing a compound reward function in the training of the motion controller, the motion controller can reasonably configure the tilt angle and rotational speed of each thruster based on the desired thrust target. Inspired by the APF method, we propose a virtual force-based planning controller. The planning controller consists of several independent action networks, and each action network is trained to generate an action-specific thrust target component. By combining thrust target components from several action networks, complex motion planning for robots can be achieved. The modular design of the controller makes the design of the reward function independent. It also reduces the training time of each modular network and enables each modular network to be replaced to perform different tasks. In this work, we obtained three action networks of path following, obstacle avoidance and escort through training, and realized the cooperative control for multiple ASRs system by their combination. The simulation and the experiment integrated into the multi-ASR system verified the effectiveness of the proposed formation control strategy. We believe that the proposed formation control strategy will help robots improve the efficiency of exploration in complex amphibious environments.

References

[1] C. Li, S. Guo, and J. Guo, “Study on obstacle avoidance strategy using multiple ultrasonic sensors for spherical underwater robots,” *IEEE Sensors J.*, vol. 22, no. 24, pp. 24458–24470, Dec. 2022.

[2] J. Wan, B. He, D. Wang, T. Yan, and Y. Shen, “Fractional-order PID motion control for AUV using cloud-model-based quantum genetic algorithm,” *IEEE Access*, vol. 7, pp. 124828–124843, 2019.

[3] S. Heshmati-Alamdari, G. C. Karras, P. Marantos, and K. J. Kyriakopoulos, “A robust predictive control approach for underwater robotic vehicles,” *IEEE Trans. Control Syst. Technol.*, vol. 28, no. 6, pp. 2352–2363, Nov. 2020.

[4] G. V. Lakhekar, L. M. Waghmare, and R. G. Roy, “Disturbance observer-based fuzzy adapted S-surface controller for spatial trajectory tracking of autonomous underwater vehicle,” *IEEE Trans. Intell. Vehicles*, vol. 4, no. 4, pp. 622–636, Dec. 2019.

[5] S. Heshmati-Alamdari, A. Nikou, and D. V. Dimarogonas, “Robust trajectory tracking control for underactuated autonomous underwater vehicles in uncertain environments,” *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 3, pp. 1288–1301, Jul. 2021.

[6] M. Cai, S. Wang, Y. Wang, R. Wang, and M. Tan, “Coordinated control of underwater biomimetic vehicle–manipulator system for free floating autonomous manipulation,” *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 51, no. 8, pp. 4793–4803, Aug. 2021.

[7] J. Lv et al., “Disturbance rejection control for underwater free-floating manipulation,” *IEEE/ASME Trans. Mechatronics*, vol. 27, no. 5, pp. 3742–3750, Oct. 2022.

[8] J. Lv, Y. Wang, S. Wang, X. Bai, R. Wang, and M. Tan, “A collision-free planning and control framework for a biomimetic underwater vehicle in dynamic environments,” *IEEE/ASME Trans. Mechatronics*, vol. 28, no. 3, pp. 1415–1424, Jun. 2023.

[9] J. Bak, Y. Moon, J. Kim, S. Mohan, T. Seo, and S. Jin, “Hovering control of an underwater robot with tilting thrusters using the decomposition and compensation method based on a redundant actuation model,” *Robot. Auto. Syst.*, vol. 150, Apr. 2022, Art. no. 103995.

[10] C. Li and S. Guo, “Adaptive multi-mode switching strategy for the spherical underwater robot with hybrid thrusters,” *Adv. Eng. Informat.*, vol. 55, Jan. 2023, Art. no. 101845.

[11] K. B. Knudsen, M. C. Nielsen, and I. Schjølberg, “Deep learning for station keeping of AUVs,” in *Proc. Oceans*, Oct. 2019, pp. 1–6.

[12] T. Zhang, R. Tian, C. Wang, and G. Xie, “Path-following control of fish-like robots: A deep reinforcement learning approach,” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 8163–8168, 2020.

[13] J. Zheng, T. Zhang, C. Wang, M. Xiong, and G. Xie, “Learning for attitude holding of a robotic fish: An end-to-end approach with sim-to-real transfer,” *IEEE Trans. Robot.*, vol. 38, no. 2, pp. 1287–1303, Apr. 2022.

[14] X. Liu, S. S. Ge, C.-H. Goh, and Y. Li, “Event-triggered coordination for formation tracking control in constrained space with limited communication,” *IEEE Trans. Cybern.*, vol. 49, no. 3, pp. 1000–1011, Mar. 2019.

[15] R. An, S. Guo, Y. Yu, C. Li, and T. Awa, "Task planning and collaboration of jellyfish-inspired multiple spherical underwater robots," *J. Bionic Eng.*, vol. 19, no. 3, pp. 643–656, May 2022.

[16] Y. Zhao, Y. Ma, and S. Hu, "USV formation and path-following control via deep reinforcement learning with random braking," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5468–5478, Dec. 2021.

[17] T. Zhang, Y. Li, S. Li, Q. Ye, C. Wang, and G. Xie, "Decentralized circle formation control for fish-like robots in the real-world via reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 8814–8820.

[18] H. Yin, S. Guo, and M. Liu, "A virtual linkage-based dual event-triggered formation control strategy for multiple amphibious spherical robots in constrained space with limited communication," *IEEE Sensors J.*, vol. 22, no. 13, pp. 13395–13406, Jul. 2022.

[19] S. Gu et al., "Communication and cooperation for spherical underwater robots by using acoustic transmission," *IEEE/ASME Trans. Mechatronics*, vol. 28, no. 1, pp. 292–301, Feb. 2023.

[20] C. Li and S. Guo, "Characteristic evaluation via multi-sensor information fusion strategy for spherical underwater robots," *Inf. Fusion*, vol. 95, pp. 199–214, Jul. 2023.

He Yin (Member, IEEE) received the B.Eng. and M.Eng. degrees in biomedical engineering from the Beijing Institute of Technology, Beijing, China, in 2017 and 2019, respectively, where he is currently pursuing the Ph.D. degree.

His current research interests include amphibious robot and multirobot cooperation.

Shuxiang Guo (Fellow, IEEE) is currently a Chair Professor with the Department of Electronic and Electrical Engineering, Southern University of Science and Technology, Shenzhen, Guangdong, China. He is also a Chair Professor with the Key Laboratory of Convergence System and Healthcare Technology Medical Engineering, Beijing Institute of Technology, Beijing, China. His current research interests include biomimetic underwater robots, medical robot systems for minimal invasive surgery, micro catheter systems, micropump, and smart materials (SMA, IPMC) based on actuators.

Prof. Guo has a fellowship of the Engineering Academy of Japan.

Ao Li (Student Member, IEEE) received the B.S. degree from the School of Automation, Beijing Institute of Technology, Beijing, China, in 2017, and the M.S. degree in biomedical engineering from the Beijing Institute of Technology, in 2019, where she is currently pursuing the Ph.D. degree in biomedical engineering.

Her current research interests include underwater robot, robot control, and hydrodynamics.

Liwei Shi (Member, IEEE) received the B.S. degree in mechanical engineering from Harbin Engineering University, Harbin, China, in 2006, and the M.S. and Ph.D. degrees in intelligent machine system engineering from Kagawa University, Takamatsu, Japan, in 2009 and 2012, respectively.

He was a Postdoctoral Researcher with Kagawa University, from 2012 to 2013. He is currently an Associate Professor with the School of Life Science, Beijing Institute of Technology, Beijing, China. His research interests include biomimetic robots, amphibious robots, and surgical robots.

Meng Liu (Member, IEEE) received the B.S. degree in control theory and applications from the Beijing Institute of Technology, Beijing, China, in 2015, where he is currently pursuing the M.S. degree in biomedical engineering with the Key Laboratory of Convergence Medical Engineering System and Healthcare Technology.

His current research interests include trajectory tracking and avoiding obstacle on amphibious spherical robot.