Applied Ocean Research 110 (2021) 102590

Contents lists available at ScienceDirect

Applied Ocean Research

journal homepage: www.elsevier.com/locate/apor

Adaptive and extendable control of unmanned surface vehicle formations using distributed deep reinforcement learning

Shuwu Wang [a,b,c,d], Feng Ma [c,d], Xinping Yan [a,c,d], Peng Wu [b], Yuanchang Liu [b,*]

[a] School of Energy and Power Engineering, Wuhan University of Technology, Wuhan, PR China
[b] Department of Mechanical Engineering, University College London, Torrington Place, WC1E 7JE, UK
[c] Intelligent Transport System Research Center, Wuhan University of Technology, Wuhan, PR China
[d] National Engineering Research Center of Water Transportation Safety (WTSC), Wuhan, PR China

ARTICLE INFO

Keywords:
Unmanned surface vehicles (USVs)
USV formation control
Deep reinforcement learning
Deep deterministic policy gradient (DDPG)
Extendable reinforcement learning

ABSTRACT

Future ocean exploration will be dominated by a large-scale deployment of marine robots such as unmanned surface vehicles (USVs). Without the involvement of human operators, USVs exploit oceans, especially the complex marine environments, in an unprecedented way with an increased mission efficiency. However, current autonomy level of USVs is still limited, and the majority of vessels are being remotely controlled. To address such an issue, artificial intelligence (AI) such as reinforcement learning can effectively equip USVs with high-level intelligence and consequently achieve full autonomous operation. Also, by adopting the concept of multi-agent intelligence, future trend of USV operations is to use them as a formation fleet. Current researches in USV formation control are largely based upon classical control theories such as PID, backstepping and model predictive control methods with the impact by using advanced AI technologies unclear. This paper, therefore, paves the way in this area by proposing a distributed deep reinforcement learning algorithm for USV formations. More importantly, using the proposed algorithm USV formations can learn two critical abilities, i.e. adaptability and extendibility that enable formations to arbitrarily increase the number of USVs or change formation shapes. The effectiveness of algorithms has been verified and validated through a number of computer-based simulations.

1. Introduction

Our oceans are the least-explored region of planet Earth. Several world leading projects are making profound contributions to a better understanding of oceans. For example, a Defence Advanced Research Projects Agency (DARPA) funded project, 'Ocean of Things', has been proposed to enable persistent maritime situational awareness over large ocean areas by deploying thousands of small, low-cost floats that could form a distributed sensor network (Waterston et al., 2019). By equipping floats with a suite of commercially available sensors, important environmental data such as ocean temperature, sea state, salinity and location, can be collected in real-time and transmitted via satellite to a cloud network for storage and real-time analysis.

It can be seen that current trend for next generation ocean exploration is towards automated and intelligent operation in extreme and harsh maritime environments. Such an ambition will be largely underpinned by recent advances in robotics and artificial intelligence (AI). Marine robots, such as unmanned surface vehicles (USVs), will play an increasingly key role in the near future and this role will continually expand and become more challenging as we extend into deeper, remote and hostile marine environments. However, current autonomy level of USVs remains relatively low as most vessels adopt remote control model. A real-time on-board intelligence is still in absence and requires further investigations.

Currently, one of the promising approaches to increase USVs' autonomy level is to deploy vessels as a formation fleet with multiple USVs simultaneously undertaking missions in a cooperative manner. Such an approach can achieve a shared autonomy, where each USV makes its own contribution towards a complex mission that requires sophisticated operations. By decomposing missions into several sub-missions that are not beyond each USV's capacity, an improved mission execution efficiency can be achieved together with other benefits such as increased mission areas and fault tolerance capability.

Central to the deployment of USV formations is the design and

* Corresponding author.
E-mail address: yuanchang.liu@ucl.ac.uk (Y. Liu).

https://doi.org/10.1016/j.apor.2021.102590
Received 17 August 2020; Received in revised form 5 January 2021; Accepted 17 February 2021
Available online 6 March 2021

development of a robust formation control strategy, which ensures a formation to reach a target point or follow a predefined trajectory by retaining a formation shape (triangular, circular or linear shapes). Several studies have been actively undertaken in this area by employing classical control strategies such as model predictive control (MPC), backstepping control and adaptive control methods. For example, Liu et al. (2018) proposed an incremental predictive control method for USV formation control and consensus. Issues of network delays and uncertainties were specifically addressed by using an incremental observer and predictive controller. Qin et al. (2017) used the sliding mode control method to solve USV formation problem with a specific aim on resolving underactuated issues. Innovatively, high-level decision-making capabilities such as task allocation and motion planning have been integrated into formation control algorithm to form a holistic hierarchical control framework. Similarly, Liang et al. (2019) developed a swarm centre position guidance algorithm using neural networks to enable each USV within a formation to follow a desired path, and to guarantee path following errors converge to a small neighbourhood of origin. Although these conventional control methods can achieve a good control performance, most of these approaches require sophisticated calculation on analytical solutions yielding a non-linear control law which is difficult to implement on practical USV platforms.

Machine learning, especially reinforcement learning (RL), can provide new insight into formation control. With the underlying concept that training an agent to learn an optimal policy via a trail-and-error manner, a high-level intelligence can be generated and applied upon many practical applications. Successful implementations of RL includes the design of AlphaGo (Silver et al., 2016) and the most recent breakthrough in playing StarCraft II (Vinyals et al., 2019). Besides, many researchers are committed to applying RL to the control of single-agent such as USV, autonomous underwater vehicle (AUV) and unmanned aerial vehicle (UAV). Wu et al. (2020) proposed a duelling deep Q-network based method for the autonomous navigation and obstacle avoidance of USVs. Sun et al.(2020) innovated an optimized sample pools and average motion critic network based DDPG algorithm for the path following control of AUVs. An actor-critic reinforcement learning algorithm is proposed by Ma et al. (2018) for the obstacle avoidance of UAVs control in continuous spaces. Although the RL performs well in the control of single-agent, some other issues (state-space explosion, the adaptability and extendibility of learned policy) need to be resolved for the formation control.

The application of RL in formation control is also evitable. Wen et al. (2017) proposed a fuzzy logic based RL control method for multi-agent formation control. Issues such as unknown dynamic and inherent nonlinearity that are difficult to be addressed using conventional control methods were well addressed. Zuo et al., (2010) studies the formation control of multiple robots using RL and a self-learning capability can be realised to initiate complex tasks execution in various environments. A $GQ(\lambda)$ algorithm was proposed by Knopp et al. (2017) for a fleet of human robot to achieve a smooth path following performance while the formation is being controlled.

In terms of using RL for USV formation control, limited studies have been carried out. Most of present USV related literature focus on applying RL for single vehicle path following control. Woo et al. (2019) proposed a deep reinforcement learning (DRL)-based controller for path following of a USV with a new Markov decision process (MDP) model for USVs designed. Wu et al. (2020) designed a new RL method for autonomous navigation and obstacle avoidance using a duelling deep Q-network. Zhou et al. (2019) innovated a new deep Q-network based control structure for USV formations with advanced collision avoidance ability been tested. However, a discretised action, i.e. a USV cannot take a continuous adjustment in heading angle and speed, was adopted, which limits the accuracy of real-time control.

By summarising current studies in using RL for USVs control especially for USV formation control, research gaps remain as follows: 1) the majority of work is investigating RL based control for a single USV with limited studies looking into USV formation control; 2) most of the research focuses on only one USV formation shape without adaptability and extendibility for different USV formation, i.e. the change of the number of vessels in a formation or a change of formation shapes requires a redesign of formation scheme; 3) the training of RL algorithm for formations is not adaptive and extendable, i.e. any changes of USV formation (number of vehicles or shape) require a redesign of RL networks and a rerun of training process. It should be noted that the adaptability and extendibility of a USV formation are of great significance. By bringing a new concept of modular design, it would become advantageous if a trained and stable RL agent (or a neural network) is readily to be applied onto any newly added USVs when a formation needs to make any change. Such a plug-and-play capability will greatly benefit the operation of USV formations in complex and demanding scenarios.

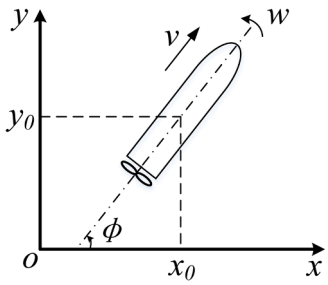

Fig. 1. USV coordination system.

Based upon the discussions above, a new adaptive and extendable control strategy for USV formations has been proposed in this paper using distributed DRL. The main contributions are: 1) a new leader-follower based USV formation method has been proposed using the architecture of distributed DRL; 2) through a novel design of USV formation structures, a great flexibility can be achieved in a way that the number of vessels in a formation or the formation shape can be adjusted by modifying associated formation parameters; 3) to enable the adaptability and extendibility of formation control, a new USV formation MDP that facilitates the reusing of control strategy has been proposed. Extensive computer-based simulations have been conducted to validate the performance of the proposed algorithms.

The rest of the paper is organised as follows. Section 2 provides a fundamental introduction to key knowledge underpinning the work in this paper such as USV dynamic model, USV formation control strategies and basics in RL. Section 3 specifically introduces the designed new algorithms for adaptive and extendable USV formation control based upon DDPG. Section 4 presents a set of simulation results to validate the algorithms. Section 5 concludes the paper and discusses future work.

2. Problem formulation

2.1. USV dynamic model

In this work, the motion of a USV is computed using the classical approach with theory of rigid body dynamics and kinematics. It should be noted that the conventional dynamic model of a USV belongs to 6 degrees of freedom (6 DoF), and in this study without losing generality, a USV's motion is assumed to be restricted to the horizontal plane, disregarding pitch and roll motions to maintain the complexity of the model at a reasonable level. Considering a coordination system as shown in Fig. 1, the motion of a USV can be written as:

$$\begin{cases} x_{t+1} = x_t + v_{t+1} * T * cos\phi_{t+1} \\ y_{t+1} = y_t + v_{t+1} * T * sin\phi_{t+1} \\ v_{t+1} = v_t + a_t * T \\ \phi_{t+1} = \phi_t + \omega_t * T \end{cases} \tag{1}$$

where x_t, y_t are the position of a USV at time t, v_t, ϕ_t are the speed and

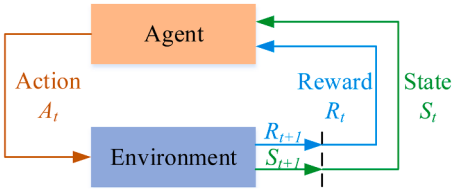

Fig. 2. The agent-environment MDP interaction framework.

heading angle of a USV at time t, a_t, ω_t, as the control inputs for a USV, are the acceleration and angular velocity at time t, respectively, T is the sampling time for a system.

2.2. USV formation control background

To achieve a USV formation control, three main strategies have been proposed including leader-follower, virtual structure and behaviour-based methods. A detailed review in formation control can be found in (Liu and Bucknall, 2018). In the leader-follower control approach, one vehicle is regarded as the group leader with full access to the overall navigation information and works as the reference vehicle in the formation. Apart from the leader vehicle, other vehicles in the formation are viewed as followers. Followers operate under the guidance of the leader with the primary aim being retention of the formation shape by maintaining the desired distance from and pose angle to the leader. The virtual structure (VS) as defined in this context is a collection of elements (unmanned vehicles), which maintain a rigid geometric relationship to each other and a frame of reference. By treating the formation shape as a VS or a rigid body, the formation is maintained by minimising the position error between the VS and actual formation position. Behaviour-based formation control solves the formation control problem by using a hybrid vector-weighted control function, which is able to generate the control command based upon various kinds of formation missions.

2.3. Fundamentals in reinforcement learning (RL)

In this section, the rationale of RL will be discussed. The MDP which is normally used for reinforcement environment modelling will be first introduced and then followed by the discussion on one of two fundamental RL algorithms – tabular based learning algorithms and approximation based learning algorithms. At last, policy gradient and deep deterministic actor-critic algorithms are introduced.

2.3.1. Markov decision process (MDP)

A MDP describes an environment for learning, in which a goal can be learned via continuous interactions between an agent and the environment. More specifically, an MDP can be represented using a 4-element tuple:

$$M = [s, a, r, p] \tag{2}$$

where $s = s_1, s_2, \ldots, s_t, s_{t+1}$ represents the dynamic environment with a finite set of states with s_t denoting the state at time t. $a = a_1, a_2, \ldots, a_t, a_{t+1}$ represents the actions executed by an agent and a_t denotes the taken action at time t. r is the reward function with $\gamma \in [0, 1]$ being the discount factor which determines the present value of future rewards with discounting. p is the transition probability function expressed as:

$$p_{ss'}^{a} = P[s_{t+1} = s' | s_t = s, a_t = a] \tag{3}$$

The interaction between an agent and an environment is shown in Fig. 2. The agent, i.e., a learner and decision maker, selects an action a_t with observed environment state s_t; the environment, in response to the actions taken by the agent, updates its state to s_{t+1} and returns an immediate reward r_{t+1} to the agent (Kaelbling et al., 1996). The selection of action is bounded by a policy ($\pi(a|s)$), which is a mapping from states to probabilities of selecting each possible action. Given a policy π, the state-value function of a state s can be calculated as:

$$V_\pi(s) = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right] \tag{4}$$

Similarly, an action-value function for a policy π can be calculated as:

$$Q_\pi(s, a) = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right] \tag{5}$$

Solving a RL problem is to find an optimal policy π^* to achieve a maximal accumulated discounted reward, which can lead to either an optimal state-value function or an optimal action-value function as expressed as:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} V_\pi(s) = \underset{\pi}{\operatorname{argmax}} Q_\pi(s, a) \tag{6}$$

2.3.2. Tabular and approximation based learning algorithms

The core to RL is the calculation of value functions (state-value function or action-value function) given the specific problem to be investigated. When state and action spaces have relatively small dimension, tabular based RL algorithms can be used where state-value or action-value are stored in tables. One of the main advantages of tabular methods is exact optimal value function and optimal policy can always be found (Sutton and Barto, 1998). Typical tabular based learning algorithms include dynamic programming, Monte Carlo methods and temporal-difference (TD) learning. Based upon the criteria whether a model of an environment is needed, dynamic programming can be categorised as model-based algorithms whilst Monte Carlo methods and TD learning are model-free algorithms. Q-learning as one of the TD learning algorithms, proposed by (Watkins and Dayan, 1992), is one of the early breakthroughs in RL (Sutton and Barto, 1998).

To address the high complexity of the value function calculation when large state spaces exist, approximation based RL algorithms are proposed. The essence of such algorithms is to apply function approximation to value functions so that an approximated mapping from states (or state-action pairs) to value functions can be established to assist with exploring an optimal policy (Sutton and Barto, 1998). It should be noted that by using the approximation based learning algorithm, not only the problem with large state spaces can be addressed, but the capability of dealing with unknown states will also be enabled, which is particularly intriguing for practical applications.

Common function approximation methods include linear combination, kernel-based approximation, memory-based approximation and neural networks. Deep Q Network (DQN) is first proposed by Mnih et al. (2015). Similar to Q-learning, DQN is also an off-policy algorithm. The main components of DQN algorithm are Q-network with parameters θ, target network with parameters θ^-, loss function and replay memory. Note that the two neural networks share the same structure (Mnih et al., 2015). There are three improvements in DQN algorithm compared to tabular Q-learning. First, DQN uses deep neural networks to approximate the action-value function. Therefore, DQN algorithm can be applied to large or continuous state space problems without the need of Q table. Second, DQN utilises experience replay to enhance the learning process. The main role of experience replay is to overcome the problem of correlated data and non-stationary distribution of empirical data by training randomly from previous state transitions (experiences). Experience replay has the advantages of high data utilization because individual samples can be used multiple times. In addition, the experience replay with mini-batches breaks the correlation of consecutive samples which can lead to large variance in network parameters. Third, DQN employs two networks with a target network providing fixed targets. The parameters θ in Q-network keep updating in each time step during

the training process, while θ^{-} update periodically. Such an update strategy improves the training stability.

2.3.3. Policy gradient and the deep deterministic actor-critic algorithm

Typical RL algorithms such as tabular Q learning (Chen et al., 2019) and DQN algorithms belong to action-value methods, i.e. the learning process is achieved in a way that values of actions are first learned, and actions are selected based upon estimated action values (Sutton and Barto, 1998). By integrating a function approximation strategy (such as the DQN), problems with a high dimension of state space can be well addressed. However, when a continuous action space exists, the iterative calculation of value functions becomes computational expensive with a weak guarantee of convergence making it inappropriate to apply value function based learning algorithms. Such a problem is of special importance for USV formation control, where a continuous and smooth control is always preferred.

To properly address these issues, policy gradient based learning methods have been proposed and well-studied. Similar to parameterise a value function, a policy can be parameterised as:

$$\pi(as, \boldsymbol{\theta}) = \Pr\{a_t = a | s_t = s, \theta_t = \boldsymbol{\theta}\} \tag{7}$$

where $\boldsymbol{\theta} \in \Re^{d'}$ is the policy's parameter vector. $\Re$ is the real domain and $\boldsymbol{d}'$ is the dimension of $\boldsymbol{\theta}$. A scalar performance measure can then be defined as $J(\boldsymbol{\theta})$. Following the gradient of the scalar performance measure, a gradient ascent can be performed to maximise $J(\boldsymbol{\theta})$ as:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \nabla J(\boldsymbol{\theta}_t) \tag{8}$$

where $\alpha \in [0, 1]$ is the learning rate. $\nabla J(\boldsymbol{\theta}_t)$ is a stochastic estimate whose expectation approximates the gradient of the performance measure. In an episodic case, $J(\boldsymbol{\theta})$ can be defined as:

$$J(\boldsymbol{\theta}) = \sum_{s \in S} d^{\pi}(s) V^{\pi}(s) = \sum_{s \epsilon S} d^{\pi}(s) \sum_{a \in A} \pi_{\boldsymbol{\theta}}(a|s) Q^{\pi}(s, a) \tag{9}$$

where $d^{\pi}(s)$ is the stationary distribution of Markov chain for $\pi_{\boldsymbol{\theta}}$ and $\nabla J(\boldsymbol{\theta})$ can thus be written as:

$$\nabla J(\boldsymbol{\theta}) \propto \sum_{s \epsilon S} d^{\pi}(s) \sum_{a \in A} Q^{\pi}(s, a) \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a|s) \tag{10}$$

It can be seen that policy gradient method includes two important components, i.e. the policy model ($\pi_{\boldsymbol{\theta}}(a|s)$) and the value function ($Q^{\pi}(s, a)$). Several methods, such as REINFORCE (Sutton and Barto, 2018) and its variations, have been proposed to use a sample return to estimate the value function. To improve the learning performance, a better estimation of the value function can assist with the policy update, and this forms the core of actor-critic training method for policy gradient. By parameterising the value function, the actor-critic method works in a way that the critic updates the value function parameters whereas the actor updates the policy parameters in the direction suggested by the critic.

The policy to be updated can be either stochastic or deterministic. A stochastic policy is often represented as a set of conditional probability distributions, which return a distribution of actions to take at each state. The stochastic policy is better suited in an uncertain environment, where exploration is favoured to have an optimal policy update result. However, this is always carried out at the cost of high computational complexity. To address this issue, a deterministic policy has been proposed and it explicitly depicts a mapping ($\mu_{\boldsymbol{\theta}}$, $\boldsymbol{\theta} \in \Re^{\boldsymbol{m}}$) from state to action: $S \rightarrow A$. Following the same rule in stochastic policy gradient, a new algorithm named deterministic policy gradient (DPG) (Silver et al., 2014) has been proposed using the actor-critic method. The performance measure for DPG can be defined as:

$$J(\boldsymbol{\theta}) = \int_S \rho^{\mu}(s) Q(s, \mu_{\boldsymbol{\theta}}(s)) ds \tag{11}$$

where $\rho^{\mu}(s)$ is the state distribution and the gradient of the measure $J(\boldsymbol{\theta})$ can be calculated as:

$$\nabla J(\boldsymbol{\theta}) = \int_S \rho^{\mu}(s) \nabla_a Q^{\mu}(s,a) \nabla_{\boldsymbol{\theta}} \mu_{\boldsymbol{\theta}}(s)|_{a=\mu_{\boldsymbol{\theta}}(s)} ds = E_{s \sim \rho^{\mu}} \left[\nabla_a Q^{\mu}(s,a) \nabla_{\boldsymbol{\theta}} \mu_{\boldsymbol{\theta}}(s)|_{a=\mu_{\boldsymbol{\theta}}(s)} \right] \tag{12}$$

To ensure a sufficient and satisfactory exploration for deterministic policy gradient, an off-policy learning strategy is adopted so that the training trajectories are generated by a stochastic policy $\beta(a|s)$ and the new performance measure and its gradient can be expressed as:

$$J_{\beta}(\boldsymbol{\theta}) = \int_S \rho^{\beta}(s) Q^{\mu}(s, \mu_{\boldsymbol{\theta}}(s)) ds \tag{13}$$

$$\nabla J(\boldsymbol{\theta}) = E_{s \sim \rho^{\beta}} \left[\nabla_a Q^{\mu}(s, a) \nabla_{\boldsymbol{\theta}} \mu_{\boldsymbol{\theta}}(s)|_{a=\mu_{\boldsymbol{\theta}}(s)} \right] \tag{14}$$

By combining the DPG and DQN, DDPG (Lillicrap et al., 2015) is proposed as a model-free off-policy actor-critic algorithm (pseudocode is shown in Algorithm 1). By retaining the feature of DQN that is stabilizing the learning of Q-function by experience replay and the frozen target network, DDPG extends the original discrete action space of DQN to a continuous space with the actor-critic framework while learning a deterministic policy (Wang et al., 2019). Both the critic and actor are equipped with two networks that have the same structure, which are the critic/actor networks and the target critic/actor networks, respectively. The update of parameters of these networks is similar to that of DQN, i.e. by minimising the loss between the target and estimated values. Also, to enable DDPG to have a satisfactory exploration in state and action spaces, when selecting an action from the policy $\mu_{\boldsymbol{\theta}}(s)$, noise $\mathcal{N}$ is added as:

$$a = \mu_{\boldsymbol{\theta}}(s) + \mathcal{N} \tag{15}$$

In addition, to stabilise the learning process, different to DQN where the target network stays frozen for some period of time, a soft update strategy is employed in DDPG for parameters update for both critic and actor networks: $\boldsymbol{\theta}' \leftarrow \tau\boldsymbol{\theta} + (1-\tau)\boldsymbol{\theta}'$ with $\tau \ll 1$.

Algorithm 1 Deep deterministic policy gradient (DDPG) (Lillicrap et al., 2015)

Randomly initialise critic network $Q(s, a|\boldsymbol{\theta}^{Q})$ and actor $\mu(s|\boldsymbol{\theta}^{\mu})$ with weights $\boldsymbol{\theta}^{Q}$ and $\boldsymbol{\theta}^{\mu}$
Initialise target network Q' and μ' with weights $\boldsymbol{\theta}^{Q'} \leftarrow \boldsymbol{\theta}^{Q}$ and $\boldsymbol{\theta}^{\mu'} \leftarrow \boldsymbol{\theta}^{\mu}$
Initialise replay buffer R
for episode = 1, M **do**
Initialise a random process $\mathcal{N}$ for action exploration
Receive initial observation state s_1
for t = 1, T **do**
Select action $a_t = \mu(s_t\boldsymbol{\theta}^{\mu}) + \mathcal{N}_t$ according to the current policy and exploration noise
Execute action a_t and observe reward r_t and observe new state s_{t+1}
Store transition (s_t, a_t, r_t, s_{t+1}) in R
Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\boldsymbol{\theta}^{\mu'})|\boldsymbol{\theta}^{Q'})$
Update critic by minimising the loss: $L = \frac{1}{N}\sum_{i=1}(y_i - Q(s_i, a_i|\boldsymbol{\theta}^{Q}))^2$
Update the actor policy using the sampled policy gradient:
$\nabla_{\boldsymbol{\theta}^{\mu}} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a\boldsymbol{\theta}^{Q})|_{s=s_i, a=\mu(s_i)} \nabla_{\boldsymbol{\theta}^{\mu}} \mu(s\boldsymbol{\theta}^{\mu})|_{s_i}$
Update the target networks:
$\boldsymbol{\theta}^{Q'} \leftarrow \tau\boldsymbol{\theta}^{Q} + (1-\tau)\boldsymbol{\theta}^{Q'}$
$\boldsymbol{\theta}^{\mu'} \leftarrow \tau\boldsymbol{\theta}^{\mu} + (1-\tau)\boldsymbol{\theta}^{\mu'}$
end for
end for

3. Distributed deep reinforcement learning based adaptive USV formation control

3.1. MDP design for USV formation

Based upon the leader-follower strategy, a USV formation MDP can

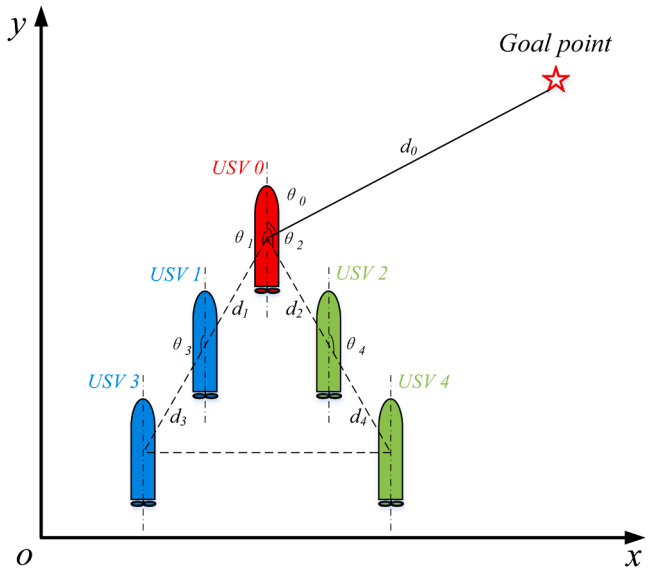

Fig. 3. USV formation configurations.

be defined. The benefits of using the leader-follower strategy include flexibility in describing the formation relationship and easy implementation and adaption of formation shapes. Also, to decouple the relationship between a leader USV and a follower USV and to have a better extendibility of formation, instead of employing a unified MDP for the whole formation, each vehicle has its own MDP. It means that the state space will not change as the number of USV increases in a formation. Besides, deep networks are employed to fit the state-action and policy functions for the continuous state space in the DDPG algorithm. The policy is iteratively updated to make USVs able to retain the formation shape while navigating towards the goal point. More specifically, as shown in Fig. 3, USVs within a formation have been grouped into three different categories as: **the leader USV** (USV0), **the follower USV on the left side** (USV 1 and 3) and **the follower USV on the right side** (USV 2 and 4). Details can be summarised as:

1) Leader USV needs to find an optimal policy to traverse to the goal point with the shortest distance;
2) Follower USVs need to identify the specific vessel it must follow and retain the formation shape by maintaining a predefined distance and angle with its following vessel. For example, as shown in Fig. 3, USV 1 needs to follow USV 0 with a preferred distance d_1 and angle θ_1.

3.1.1. State space and action space design

Based upon the categorisation of USVs within a formation, the state spaces can be defined according to the specific type of a USV. For a leader USV, with the primary aim being to reach the goal point, the state space is defined as:

$$S_{leader} = [\Delta\theta] = [\theta_0 - \varphi_0] \tag{16}$$

where θ_0 is the angle of the goal point with respect to the heading direction of the leader USV and φ_0 is the heading angle of the USV.

For follower USVs, both types (USV on the left side and USV on the right side) share the same state space as:

$$S_{follower} = \left[\frac{\Delta d_n^m}{d_{req}},\ \Delta\theta_n^m,\ \frac{\Delta v_n}{v},\ \phi_n,\ \frac{\Delta v_m}{v},\ \phi_m\right] \tag{17}$$

where $\Delta d_n^m = d_n^m - d_{req}$, d_n^m is the distance between the m^{th} and n^{th} USV (the n^{th} USV is following the m^{th} USV), d_{req} is the required formation distance between two vehicles. $\Delta\theta_n^m = \theta_n^m - \theta_{req}$ with θ_n^m being the formation angle of the n^{th} USV and θ_{req} being the required angle. $\Delta v_n = v_n - v$, with v_n being the velocity of the n^{th} USV and v is the desired velocity of the formation (the same definition of the mth USV applies to Δv_m). ϕ_n is the heading angle of the n^{th} USV (the same definition of the m^{th} USV applies to ϕ_m).

The action space can also be defined according to the specific type of USVs. For a leader USV, following a common approach of USV manoeuvring, i.e. the heading of a USV is always being adjusted while keeping the speed constant, the action space for a leader USV is defined as:

$$A_{leader} = [\omega_0] \tag{18}$$

where ω_0 is the angular velocity of leader USV.

In terms of follower USVs, following the dynamics of USVs, all followers share the same action space, which can be defined as:

$$A_{follower} = [a_n,\ \omega_n] \tag{19}$$

where a_n is the acceleration of the n^{th} USV and ω_n is the angular velocity of the n^{th} USV. Such a configuration has taken into account the fact that different to the leader USV, followers need to change their speed to travel to the desired formation positions if a formation shape is not formed or unstable.

3.1.2. USV formation rewards design

The rewards design plays a critical role in RL. According to the categories of USVs in a formation, different USV have different roles. For example, the pivoting role of a leader USV is to seek an optimised trajectory to reach the target point; whereas, followers are mainly to follow the leader vehicle and retain the formation shape. As a result of this, the reward functions in this work can be defined according to the specific type of USV. For a leader USV, the aim is to reach the goal point and when the leader USV arrives at the goal point, a zero reward will be assigned. Otherwise, the reward function is defined as:

$$r_0^\theta = -|\theta_0 - \varphi_0| \tag{20}$$

where θ_0 is the azimuth angle of the goal point with respect to the leader USV and φ_0 is the heading angle of the leader USV.

The total reward of leader USV is defined as:

$$r_{leader} = \lambda_0^\theta * r_0^\theta \tag{21}$$

where λ_0^θ is a constant which aims to define the importance of the reward r_0^θ. In this research, λ_0^θ is set to 1.0 and it can be adjusted to balance with other rewards in further researches.

For any n^{th} follower USV, the aim is to maintain the formation shape, which will be affected by two factors including formation distance and formation angle. The reward function evaluating the quality of formation distance can be defined as:

$$r_n^d = -\left|\frac{d_n^m - d_{req}}{d_{req}}\right| \tag{22}$$

where d_n^m is the distance between the m^{th} and the n^{th} USV (the n^{th} USV is following the m^{th} USV), d_{req} is the required formation distance between two vehicles.

The reward function expressing the formation angle can be defined as:

$$r_n^\theta = -|\theta_n^m - \theta_{req}| \tag{23}$$

where θ_n^m is the formation angle of USV n according to Fig. 3 (USV n is following USV m). θ_{req} is the required formation angle.

The total reward of the n^{th} follower USV is the weighted sum of the two reward functions including formation distance and formation angle as:

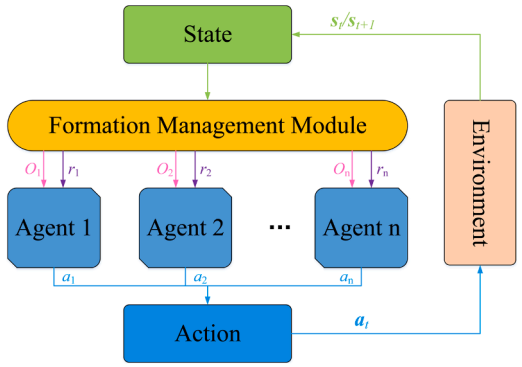

Fig. 4. The interaction of multiple USVs using deep reinforcement learning (DRL).

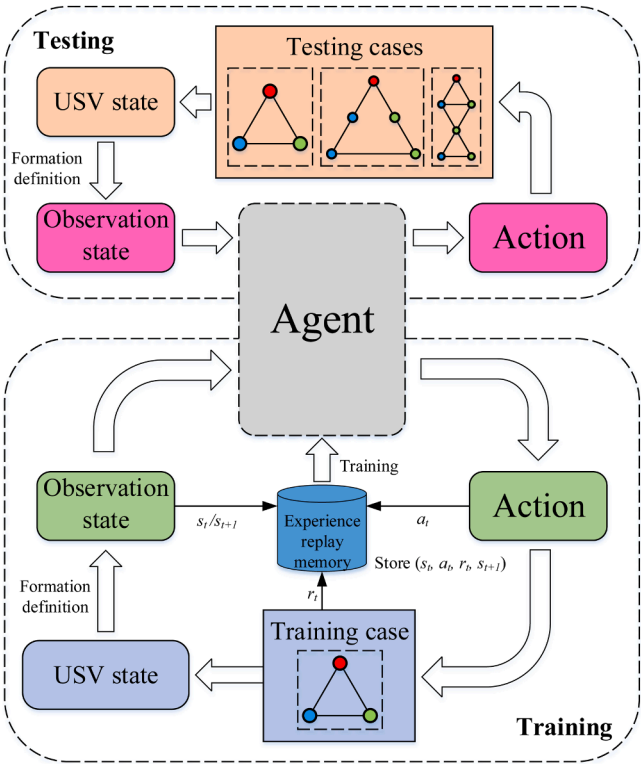

Fig. 5. Agent training and testing framework of adaptive and extendable USV formation control.

$$r_{follower} = \lambda_n^d * r_n^d + \lambda_n^\theta * r_n^\theta \tag{24}$$

where λ_n^d and λ_n^θ are constant coefficients which aim to define the importance of the reward r_n^d and r_n^θ, respectively. If the formation distance is more important than the formation angle, λ_n^d will be relatively larger than λ_n^θ to make sure the formation distance converges more quickly. Besides, λ_n^d and λ_n^θ will be used to trade off with other rewards in more complex missions. In this research, λ_n^d and λ_n^θ are equal to 1.0.

3.2. Adaptive and extendable USV formation control using deep deterministic policy gradient

In this work, an adaptive and extendable USV formation control is achieved using DDPG algorithm. Instead of carrying out a centralised training procedure, where USVs are trained as a unit, a distributed training strategy has been designed as shown in Fig. 4. Each USV (or agent as shown in Fig. 4) is equipped with its own actor-critic network. Rather than using a full observation of the environment and all the other vehicles, an observation of its own states (O_i, $i = 1, 2, \ldots, n$) can sufficiently enable a USV to learn an optimal policy by interacting with the environment using a combined action signal.

In addition, such a training strategy can largely satisfy the requirement for an extendibility for USV formations. Once the networks' parameters have been successfully updated, the trained neural network can be easily extended to other USVs if there is a requirement to increase the number of USVs in a formation. Also, because the design of the state space (explained in Section 3.1.1) does not require an explicit expression of formation shapes, using the trained networks, new formation shapes can be easily adapted, extended and formed by taking in new formation distance and angle specifications.

The specific training and testing framework is shown in Fig. 5. It is intended that by undertaking the RL training on simple USV formations (the training block in Fig. 5), USVs can directly use the learned knowledge to conduct more complex formation configurations during the testing procedures (the testing block in Fig. 5). More specifically, a triangular USV formation including all USV categories (leader, left side follower and right side follower) are involved in the training procedure, and the control policy of each USV is optimised separately, which is different from the original DDPG algorithm. In the testing phase, cases with different formation numbers and shapes are designed for validating the adaptability and extendibility of the proposed approach. With an immense benefit from the specific observation state and reward designs, USVs belonging to the same category can reuse the learned policy from training procedure.

In terms of the specific design for both actor and critic networks for the DDPG algorithm, a structure with two hidden layers (each layer containing 400 and 300 neurons respectively) has been adopted (shown in Fig. 6) based on the dimension of state and action spaces, which is

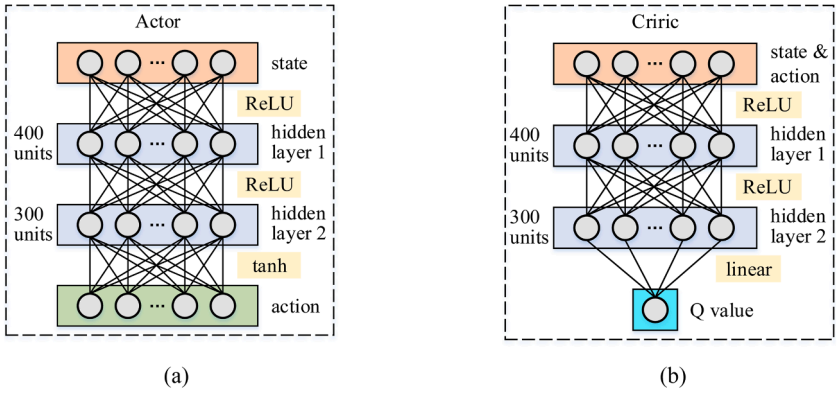

Fig. 6. Neural network design of Actor-Critic Framework. (a) and (b) represent the Actor network and the Critic network structure respectively.

Table 1
List of parameters used in deep deterministic actor-critic algorithm.

ID	Parameter	Value
1	Actor learning rate	0.0001
2	Critic learning rate	0.001
3	Soft update τ	0.001
4	Discount rate	0.99
5	Memory size	100,000
6	Batch size	128
7	Hidden layer 1	400 units
8	Hidden layer 2	300 units

sufficient to approximate the state-action and policy functions in this research. Noise $\mathcal{N}$ for action selection is constructed using Ornstein-Uhlenbeck (O-U) process to make full exploitation of state and action spaces. The O-U process can be defined as:

$$\mathcal{N}_{t+1} = \mathcal{N}_t + \theta_{OU}(\mu_{OU} - \mathcal{N}_t) + \sigma_{OU}\mathcal{N}(0,\ 1) \tag{25}$$

where $\mathcal{N}_t$ and $\mathcal{N}_{t+1}$ are the values of O-U process at time t and $t+1$, respectively. $\mathcal{N}_1$ is equal to zero. $\mathcal{N}(0,\ 1)$ is a Gaussian process with zero mean and a standard deviation of 1. The parameters for O-U process in this work are defined as $\mu_{OU} = 0,\ \theta_{OU} = 0.15,\ \sigma_{OU} = 0.2$. Other parameters for DDPG based adaptive and extendable USV formation control are listed in Table 1 with the training framework of DDPG shown in Fig. 7. All the parameters listed in Table 1 are already tuned to make sure that the training converges within acceptable time in this research.

3.3. Environment design of USV formation

The environment of the RL consists of two parts, i.e. the USV dynamic model (see Section 2.1) and the USV formation definition (see Section 3.1). Algorithm 2 depicts how the environment of the USV formation problem is formulated. Before the start of episodes, all parameters of USV formation should be configured. In each learning episode, the environment randomly samples a goal point and resets the positions and headings of USV. At each time step of an episode, the action a_t from the agent is determined at state S denoted by s_t. Then, the positions and headings are updated with Eq. (1) with the action input a_t. After that, the observation state s_{t+1} and immediate reward r_t can be calculated with Eqs. (16) and (17) and (20)-(24). This process repeats until the *Termination* is true. Note that the environment would carry out the termination of an episode if the leader USV arrives at the goal point. In this work, the tolerance for leader USV arriving at the goal point is equal to 5 m, i.e. the product of speed (5 m/s) and sampling time (1 s). In addition, termination occurs when the final targeted time step has been reached.

Algorithm 2 Environment of the USV formation problem
Set the number of USV in formation
Determine the category for each USV
Configure distance and angle of the USV formation
for episode = 1, M **do**
Randomly select a goal point and reset the positions and headings of USV
Set *Termination* = *False*
Calculate the observation state s_1 with Eqs. (16) and (17)
for$t$ = 1, T **do**
With action input a_t from the agent, at state S indexed as s_t
Get the next positions and headings of USV with Eq. (1)
Update the next observation state s_{t+1} with Eqs. (16) and (17)
Calculate the immediate reward r_t with Eqs. (20) and (24)
if leader USV arrives at goal point **then**
Termination = *True*
Break
end if
end for
Termination = *True*
end for

3.4. Agent training

The target of the USV formation task is to minimise the overall travel cost from the start point to the goal point. For the leader USV, the travel cost is the cumulative value which is the sum of the difference between the heading and the direction of goal point. With regard to the follower USV, it is the cumulative value of the weighted sum of formation distance and angle errors. Such a USV formation task is intended to train a group of agents for USV formation control with different numbers and shapes. The learning process is an episodic task. In each episode, a set of historical experiences is sampled from the replay buffer for the agents to interact with to learn policies minimising the travel cost. This process repeats until the average episode reward converges. Historical experience is collected during the agents' training procedure.

The RL agent training and policy application follow the procedure presented in Fig. 8. At the beginning of the procedure, the RL training parameters such as the learning rate and the noise of exploration, and the USV formation environment parameters such as numbers and shapes in the formation should be configured. Once the training is converged, the learned policy, i.e. the strategy of USV formation control, needs to be validated using a different set of formation tasks. In the application

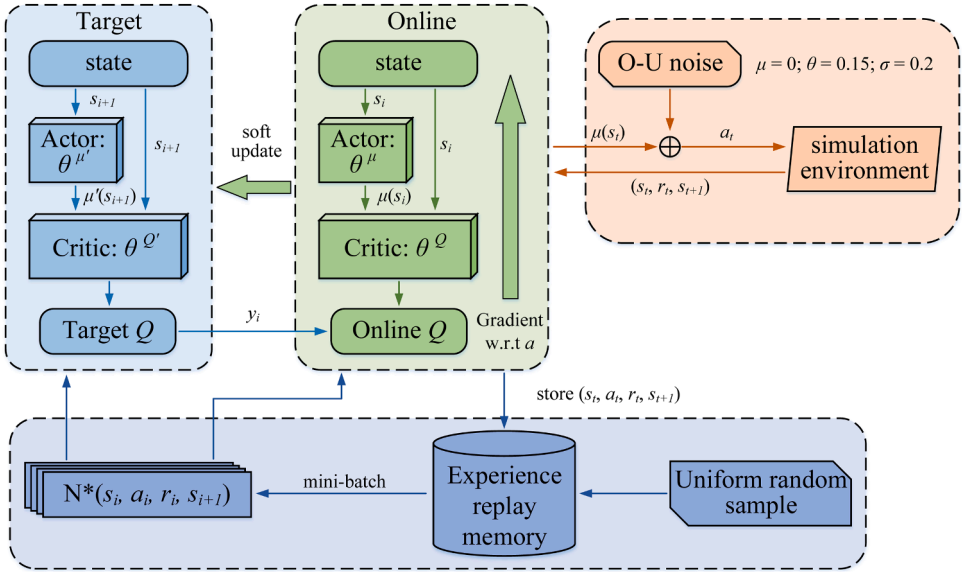

Fig. 7. The framework of deep deterministic policy gradient (DDPG) algorithm based upon actor-critic structure.

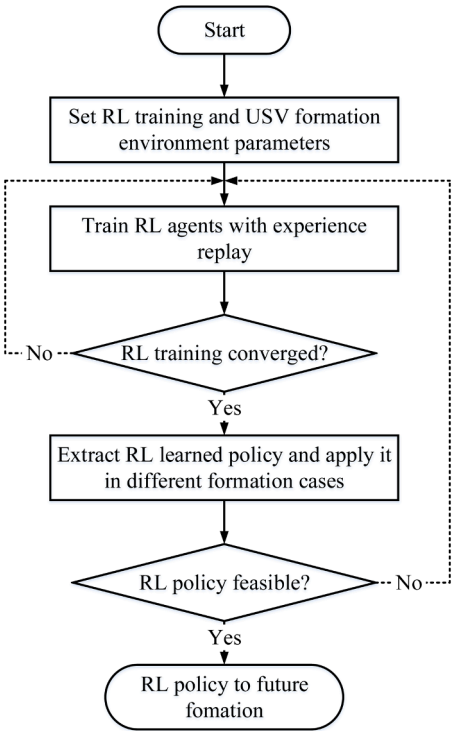

Fig. 8. Reinforcement learning (RL) agent training and policy application procedure.

phase, the learned policy needs to be tested in different formation cases. The policy can be applied to future formation when it is feasible in all test cases. Otherwise, more training episodes will be needed.

Usually, in a multi-agent system, the environment for each agent is continuously changing during the learning procedure. However, with the special state design, the environment for each USV is relatively stable in this research. For example, during the training procedure, the environment for the leader USV is always static. Although the environment for follower USVs is dynamic due to the learning of leader USV, it will not take much time for leader USV to learn a stable strategy as the task is relatively easy. Once the training of leader USV is converged, the action change of the leader USV will not be significant. It means that the environment for follower USVs will become static in a short time, which ensures that the proposed distributed training method can successfully solve the unstable environment in this research.

4. Simulation results and analysis

To verify and validate the performances of the proposed algorithms, a set of computer-based simulations have been carried out in this section. The underlying aim of this work is that by employing a distributed RL architecture, an adaptive USV formation control can be achieved. More importantly, the trained actor-critic neural networks can be directly applied onto any newly added USVs so that a USV formation can undertake missions adaptively and in an extendable way. Based upon this, intriguing simulation results in this section are presented in following ways:

1) **Demonstrating the performance of the designed distributed RL structure**. The training results of the developed distributed DDPG algorithms for a USV formation consisting of 3 vessels are presented in Section 4.1;
2) **Validating the adaptability of the proposed algorithms**. Using the trained networks in Section 4.1, simulation results that demonstrating the same USV formation can tracking different goal points are presented in Section 4.2;

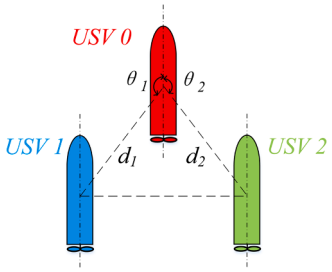

Fig. 9. USV formation configuration for reinforcement learning (RL) training.

Table 2
List of parameters used in Training.

ID	Parameter	Value
1	Max steps	200
2	Initial position of USV 0	(250, 250) m
3	Initial position of USV 1	(220, 220) m
4	Initial position of USV 2	(280, 220) m
5	Initial speed	5 m/s
6	Initial heading	90°
7	Angular velocity range of leader USV	[−4, 4] °/s
8	Angular velocity range of follower USV	[−6, 6] °/s
9	Acceleration range of follower USV	[1, 9] m/s^2
10	Distance requirements of formation	30 m
11	Angle requirements of formation	150°
12	Sampling time	s

3) **Validating the expendability of the proposed algorithms by changing the number of USVs in a formation**. A USV formation with 5 vessels is tested in Section 4.3;
4) **Validating the expendability of the proposed algorithms by changing the shape of USV formation**. A USV formation with different shapes is tested in Sections 4.4 and 4.5.

The algorithms are coded in Python with the deep neural networks built and trained using PyTorch v1.4.0.

4.1. Training results

The training of the proposed algorithms is carried out on a USV formation with the most common shape, i.e. an equilateral shape. It is anticipated that USVs can learn a generalised strategy by performing training in a simple scenario and applied the learned strategies to accommodate more complicated requirements, such as changing goal points, increasing USV numbers in a formation and varying formation shapes.

The USV formation used for training in this work is shown in Fig. 9. USV 0 is assigned as the leader USV while USV 1 and 2 are followers. With the consideration of practical operation situations, assumptions have been made that the leader USV will only make a change to its heading whereas followers can adjust both their speeds and headings. Such a configuration is to ensure that the USV formation can be trained to reach the goal point in the shortest time and well maintain its shape. The specific angular velocities and accelerations of leader and follower USVs are listed in Table 2.

The training environment has a dimension of 500 m * 500 m. The maximum step for an episode during training is set to be 200, and the positions of three USVs at the beginning of the training are (250, 250) m, (220, 220) m and (280, 200) m, respectively with the same initial speed of 5 m/s and initial heading angle of 90°. Other important training information such as the formation configuration is also shown in Table 2. In the process of training, an episode will be terminated if the number of steps has reached the maximum in one episode or the leader USV arrives at the goal point.

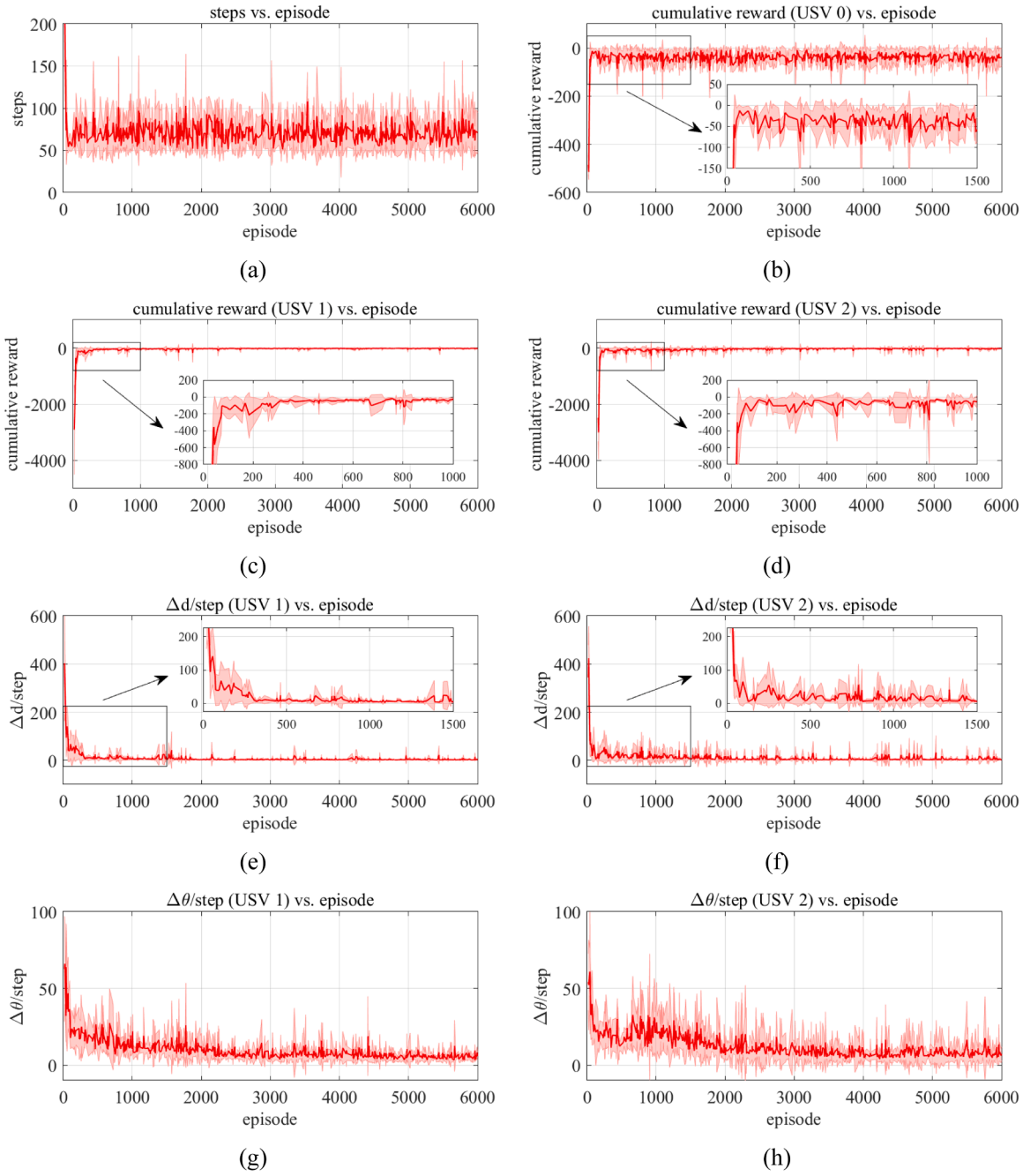

Fig. 10. Training result. (a) The steps per episode. (b) - (d) The cumulative rewards per episode of USV 0, USV 1 and USV 2, respectively. (e) - (f) The average formation distance errors per episode of USV 1 and USV2. (g) - (h) The average formation angle errors per episode of USV 1 and USV 2.

In order to have full visibility into the data, model code and parameters of the designed DRL algorithms, the randomness of the training process has been taken into account by adopting a random seed across the parameters within the built neural networks to achieve reproducibility. A total of 5 pieces of training with different seed values have been run with the results shown in Fig. 10. It can be observed that the steps of the USV formation to reach the goal point can fast converge to a stable value (Fig. 10(a)). The reward values for three different USVs (Fig. 10(b) – (d)) can also converge demonstrating that the training of the algorithms has been successful, which is further proved by formation performance metrics such as the formation distance and angle (Fig. 10(e) – (h)). Overall, the RL training of USV formation converges quickly with a reasonable policy learned thanks to the specific designs of the networks and the choice of the parameters listed in Table 1 for the DDPG algorithm.

4.2. Testing case 1

Using the trained networks for a USV formation, the learned optimal policy is directly used to test the performance of USV formation control by tracking different goal points. As shown in Fig. 11, four different testing scenarios have been configured to verify if the formation can take the correct manoeuvring actions and keep the formation shapes. Especially, in scenario 3 and 4, goal points are set in positions that are behind the starting point, which requires the formation to take a full turning. Other parameters for the USV formation used in this testing are listed in Table 3.

As shown in Fig. 11, the tracking trajectories of the formation have

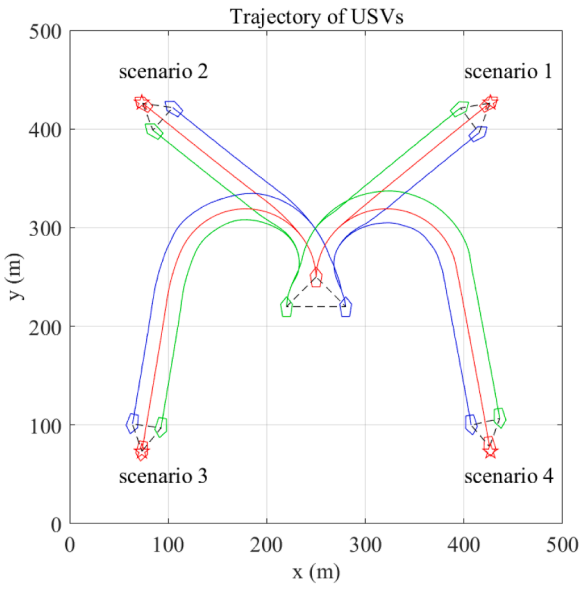

Fig. 11. Trajectories of USV formation with 4 different test scenarios in Test case 1.

Table 3
List of parameters used in USV formation Testing case 1.

USV ID	Initial position	Followed USV ID	USV Type	Formation distance	Formation angle
0	(250, 250) m	Goal point	Leader	–	–
1	(220, 220) m	0	Left	30 m	150°
2	(280, 220) m	0	Right	30 m	150°

been displayed with the red line representing the route taken by the leader USV and the green and blue lines representing the routes taken by two followers respectively. The size of formation shape at the starting point is slightly larger than the required size and it can be seen that using the trained policy the formation is able to learn to adjust its size to the predefined one and well arrived at goal points in four different scenarios. Such a performance can be further validated in the quantities evaluation results shown in Fig. 12. These values show the performance of the formation executing scenario 3. As reflected in Fig. 12(c) and (d), both the formation angle error and distance error can be well minimised close to 0 representing that a good formation shape can be maintained. The slight surge of these two values at around step 25 indicates that the formation is making a large port side turning.

4.3. Testing case 2

In this test, the extendibility of the formation by increasing the number of USVs has been verified. Two more USVs are added into the formation to form and retain a triangular formation shape. The added two USVs have been assigned the USV ID 3 and 4 as shown in Table 4 and they are navigating as follower USVs in the formation with the USVs to be followed are USV 1 and 2, respectively. The other formation parameters are listed in Table 4.

Using the same simulation environment in Testing case 1, the results revealing the trajectories of the USV formation are shown in Fig. 13. The start point is located at the centre and four different goal points (as shown as four different test scenarios) are placed at four different corners of the environment. A relatively loose formation shape is formed

Table 4
List of parameters used in USV formation Testing case 2.

USV ID	Initial position	Followed USV ID	USV Type	Formation distance	Formation angle
0	(250, 250) m	Goal point	Leader	–	–
1	(220, 220) m	0	Left	30 m	150°
2	(280, 220) m	0	Right	30 m	150°
3	(190, 190) m	1	Left	30 m	150°
4	(310, 190) m	2	Right	30 m	150°

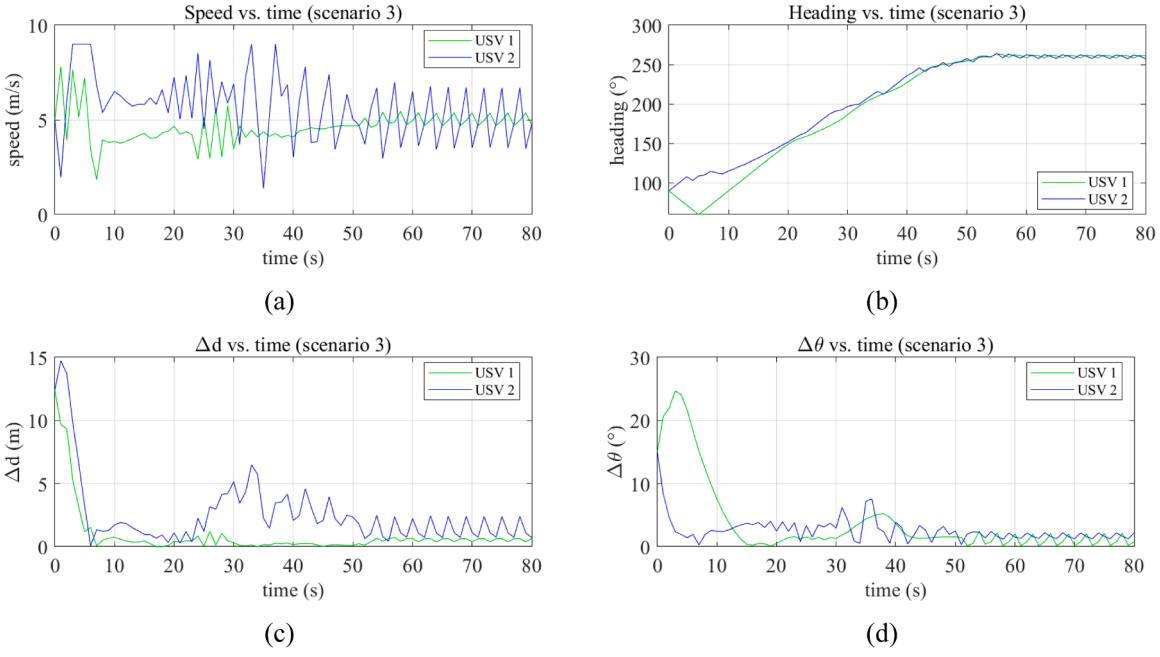

Fig. 12. Formation assessment of follower USVs in scenario 3 in Test case 1.

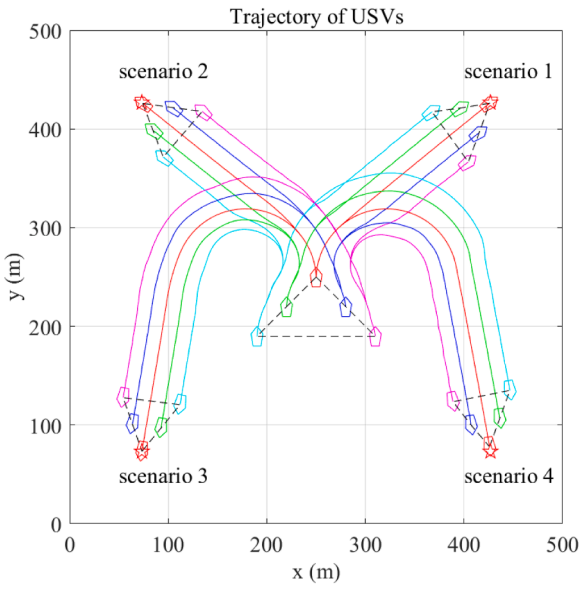

Fig. 13. Trajectories of USV formation with 4 scenarios in Test case 2.

when the formation starts travelling, and it can be observed that USVs can well navigate to keep the predefined formation distance of 30 m and formation angle of 150° and generate the desired shape. Also, such a shape can be retained along the trajectories in all four different testing scenarios. In Fig. 14, a quantitative assessment of the USV formation in scenario 3 is presented. It can be seen that four follower USVs have learned to adjust their speeds and headings (Fig. 14(a) and (b)) to perform the formation control and reduce the formation distance and angle error (Fig. 14(c) and (d)) to a small range around zero.

4.4. Testing case 3

The third test in this work is to verify the capability of using the learned policy to not only extend the number of USVs but the shape of a formation. As shown in Fig. 15, a complex double-triangular formation shape is adopted. Three more vessels (USV 3, 4 and 5) have been added with USV 3 following USV 1, USV 4 following USV 3 and USV 5 following USV 3. Based upon the relative position of a USV in a formation, neural networks are replicated accordingly, i.e. the network trained for USV 1 will be directly used for all left-side USVs and the network for USV 2 will be for right-side USVs.

The simulation results are shown in Figs. 16 and 17. Fig. 16 displays the trajectories of the USV formation while it is tracking four different goal points. Again, a good formation keeping capability can be achieved by using the trained optimal policy in a simple formation shape (in Section 4.1). Cases of different manoeuvrability including following a straight line (scenario 1), taking port and starboard side turning (scenario 2 and 4) and making a large full turn (scenario 3) have all been validated. For the most adverse action (a full turn), specific quantitative assessment is shown in Fig. 17. It can be seen that the formation distance and angle errors are able to converge to zero demonstrating that the shape of the formation can be well maintained.

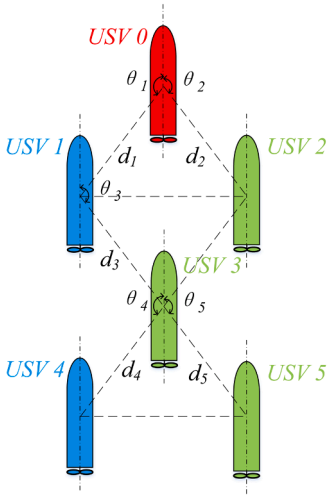

Fig. 15. The USV formation shape used in testing case 3.

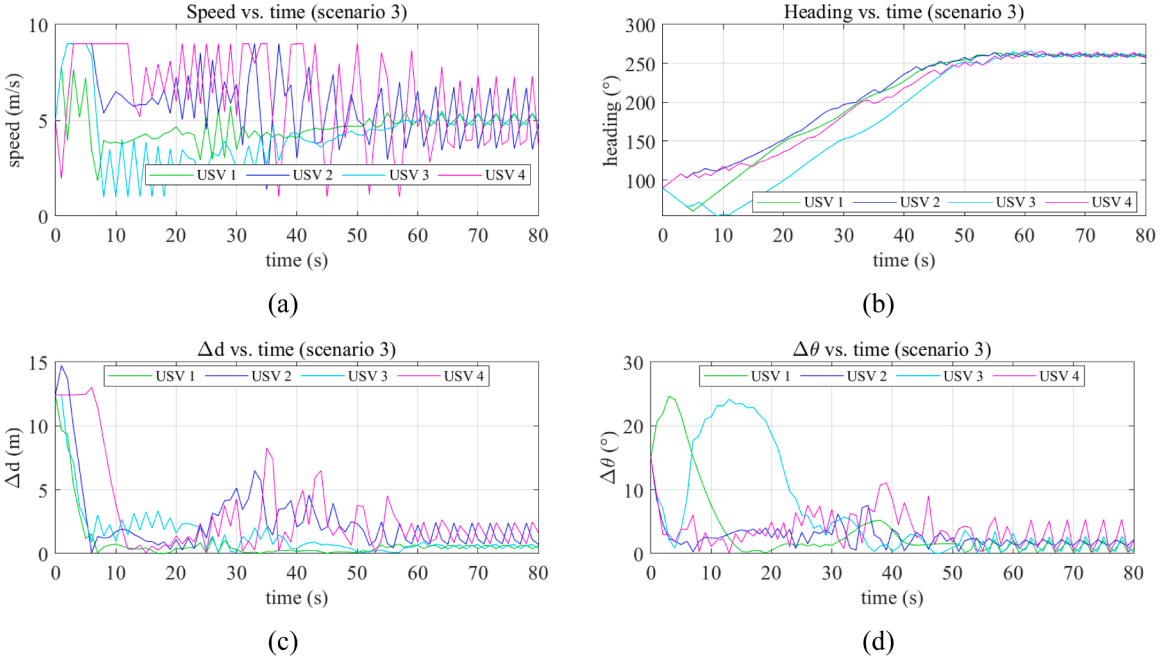

Fig. 14. Formation assessment of follower USVs in scenario 3 in Test case 2.

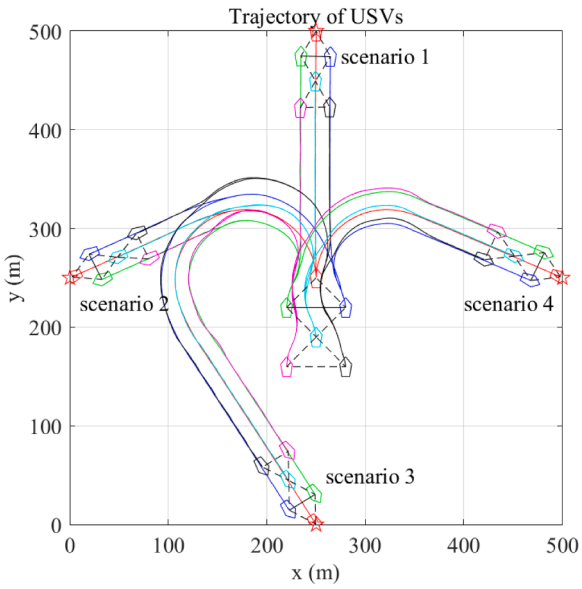

Fig. 16. Trajectories of USV formation with 4 scenarios in Test case 3.

4.5. Testing case 4

To further verify the capability of the learned policy, a more difficult testing case with new initial parameters is conducted. The initial positions of USV 3, USV 4 and USV5 are (250, 220) m, (220, 190) m and (280, 190) m, respectively. Other configurations are the same as testing case 3 as shown in Table 5. In this testing case, owing to a significant difference between the initial formation shape and the target formation shape, a good adaptability of the proposed method is therefore required and validated.

The simulation results of this testing case are shown in Figs. 18 and 19. Similarly, Fig. 18 shows the trajectories of the USV formation in four scenarios and specific quantitative assessment of scenario 3 is shown in Fig. 19. It can be seen that the formation distance and angle errors can still converge to a low level around zero, which proves that the proposed

Table 5
List of parameters used in USV formation testing case 3.

USV ID	Initial position	Followed USV ID	USV Type	Formation distance	Formation angle
0	(250, 250) m	Goal point	Leader	–	–
1	(220, 220) m	0	Left	30 m	150°
2	(280, 220) m	0	Right	30 m	150°
3	(250, 190) m	1	Right	30 m	150°
4	(220, 160) m	3	Left	30 m	150°
5	(280, 160) m	3	Right	30 m	150°

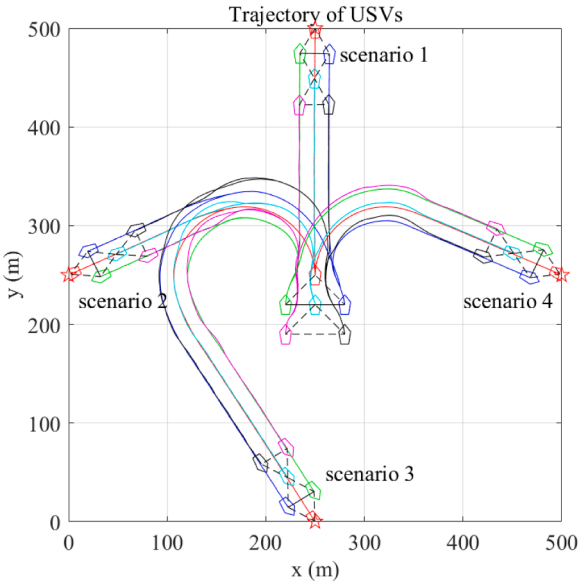

Fig. 18. Trajectories of USV formation with 4 scenarios in Test case 4.

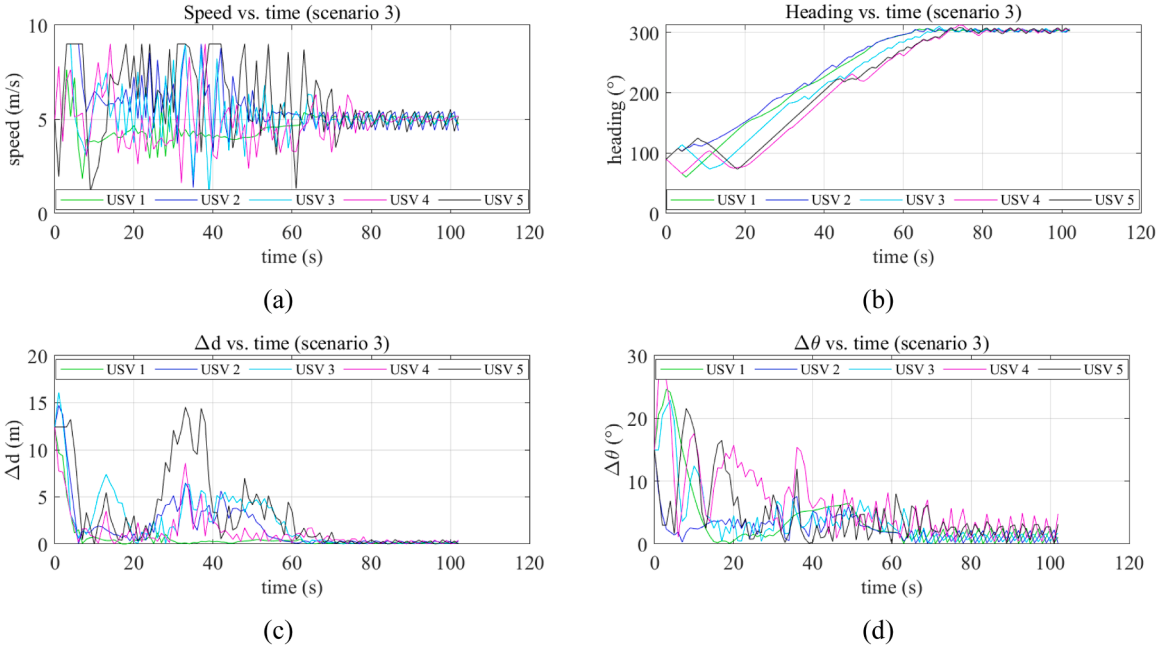

Fig. 17. Formation assessment of follower USVs in scenario 3 in Test case 3.

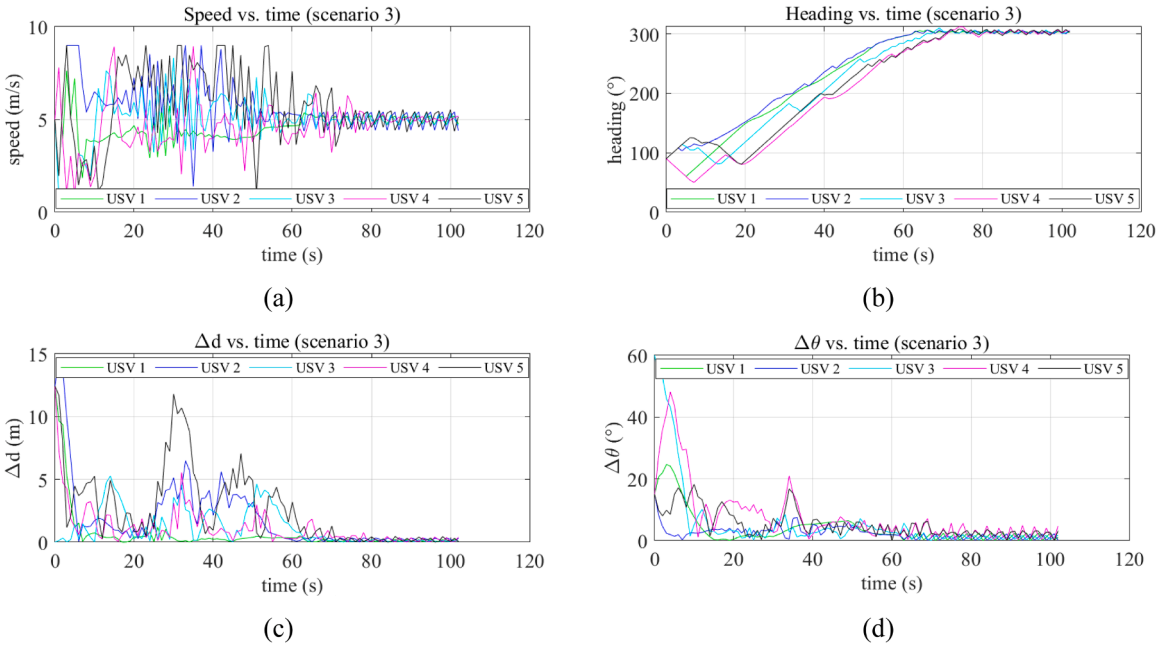

Fig. 19. Formation assessment of follower USVs in scenario 3 in Test case 4.

method is capable of maintaining the formation shape well.

Albeit only one new formation shape been tested in this section, it should be noted that by specifying the formation distance and angle parameters as shown in Table 5, arbitrary formation shape can be easily defined. By identifying which trained neural networks should be replicated onto the newly added vessels, a robust formation control can be achieved in a plug-and-play manner with the evident advantages of adaptability and extendibility.

5. Conclusions and future work

In this work, a new leader-follower based USV formation control algorithm has been developed using the distributed DRL. The special design of the USV formation structure ensures that the number of vessels in a formation or the formation shape can be parameterized. To enable adaptability and extendibility of formation control, a new USV formation MDP has been developed, which makes USVs belonging to the same category able to reuse the learned policy from DRL trainings. Such a plug-and-play functionality can potentially support USVs to undertake missions with an improved flexibility.

In terms of future work, the training of RL algorithms can be improved. Despite the advantages of DDPG algorithm especially in high dimensional action space, more advanced training algorithms such as Trust Region Policy Optimization (TRPO) and Proximal Policy Optimization (PPO). Both TRPO and PPO introduce a surrogate objective function and a Kullback-Leibler divergence constraint, guaranteeing non-decreasing long-term reward (Schulman et al., 2015, 2017). This will effectively improve training performance as well as decrease the complexity of implementation and computation.

Another important further research direction is to implement the proposed plug-and-play training manner in more complex environments. Aspects such as complicated environmental constraints including collision avoidance inside and outside the formation, impacts of meteorology and hydrology on the movement of USV and demanding mission requirements should be addressed. All of these can refer to the research on single USV (Woo and Kim, 2020; Zhou et al., 2019). The USV formation MDP has to be further customised to accommodate these challenging conditions while not exponentially increasing the computational complexity.

CRediT authorship contribution statement

Shuwu Wang: Conceptualization, Methodology, Software, Writing - original draft, Writing - review & editing. **Feng Ma:** Supervision, Software, Methodology. **Xinping Yan:** Funding acquisition, Project administration. **Peng Wu:** Methodology, Software, Writing - original draft. **Yuanchang Liu:** Conceptualization, Supervision, Funding acquisition, Project administration, Writing - original draft, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is partially supported by Royal Society (Grant no. IEC \NSFC\191633). The first author is also supported by Chinese Scholarship Council (CSC).

References

Chen, C., Chen, X.Q., Ma, F., Zeng, X.J., Wang, J., 2019. A knowledge-free path planning approach for smart ships based on reinforcement learning. Ocean Eng. 189, 106299.
Kaelbling, L.P., Littman, M.L., Moore, A.W., 1996. Reinforcement learning: a survey. J. Artif. Intell. Res. 4, 237–285.
Knopp, M., Aykın, C., Feldmaier, J., Shen, H., 2017. Formation control using GQ (λ) reinforcement learning. In: 2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN). IEEE, pp. 1043–1048.
Liang, X., Qu, X., Wang, N., Li, Y., Zhang, R., 2019. Swarm control with collision avoidance for multiple underactuated surface vehicles. Ocean Eng. 191, 106516.
Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., ... & Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint* arXiv:1509.02971.
Liu, Y., Bucknall, R., 2018. A survey of formation control and motion planning of multiple unmanned vehicles. Robotica 36 (7), 1019–1047.
Liu, Z.Q., Wang, Y.L., Wang, T.B., 2018. Incremental predictive control-based output consensus of networked unmanned surface vehicle formation systems. Inf. Sci. (Ny) 457, 166–181.
Ma, Z., Wang, C., Niu, Y., Wang, X., Shen, L., 2018. A saliency-based reinforcement learning approach for a UAV to avoid flying obstacles. Rob. Auton. Syst. 100, 108–118.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., ..., Petersen, S., 2015. Human-level control through deep reinforcement learning. Nature 518 (7540), 529–533.
Qin, Z., Lin, Z., Yang, D., Li, P., 2017. A task-based hierarchical control strategy for autonomous motion of an unmanned surface vehicle swarm. Appl. Ocean Res. 65, 251–261.
Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P., 2015. Trust region policy optimization. In: International Conference on Machine Learning, pp. 1889–1897.
Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint* arXiv:1707.06347.
Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., ..., Dieleman, S., 2016. Mastering the game of Go with deep neural networks and tree search. Nature 529 (7587), 484.
Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014, June). Deterministic policy gradient algorithms.
Sun, Y., Ran, X., Zhang, G., Wang, X., Xu, H., 2020. AUV path following controlled by modified Deep Deterministic Policy Gradient. Ocean Eng. 210, 107360.
Sutton, R.S., Barto, A.G., 1998. Introduction to Reinforcement Learning. MIT press, Cambridge. Vol. 135.
Sutton, R.S., Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.
Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., ..., Oh, J., 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature 575 (7782), 350–354.
Wang, W.Y., Ma, F., Liu, J., 2019. Course tracking control for smart ships based on a deep deterministic policy gradient-based algorithm. In: 2019 5th International Conference on Transportation Information and Safety (ICTIS). IEEE, pp. 1400–1404.
Waterston, J., Rhea, J., Peterson, S., Bolick, L., Ayers, J., Ellen, J., 2019. Ocean of things: affordable maritime sensors with scalable analysis. In: OCEANS 2019-Marseille. IEEE, pp. 1–6.
Watkins, C.J., Dayan, P., 1992. Q-learning. Mach. Learn. 8 (3–4), 279–292.
Wen, G., Chen, C.P., Feng, J., Zhou, N., 2017. Optimized multi-agent formation control based on an identifier–actor–critic reinforcement learning algorithm. IEEE Trans. Fuzzy Syst. 26 (5), 2719–2731.
Woo, J., Kim, N., 2020. Collision avoidance for an unmanned surface vehicle using deep reinforcement learning. Ocean Eng. 199, 107001.
Woo, J., Yu, C., Kim, N., 2019. Deep reinforcement learning-based controller for path following of an unmanned surface vehicle. Ocean Eng. 183, 155–166.
Wu, X., Chen, H., Chen, C., Zhong, M., Xie, S., Guo, Y., Fujita, H., 2020. The autonomous navigation and obstacle avoidance for USVs with ANOA deep reinforcement learning method. Knowl. Based Syst., 105201
Zhou, X., Wu, P., Zhang, H., Guo, W., Liu, Y., 2019a. Learn to navigate: cooperative path planning for unmanned surface vehicles using deep reinforcement learning. IEEE Access 7, 165262–165278.
Zhou, Z., Zheng, Y., Liu, K., He, X., Qu, C., 2019b. A real-time algorithm for USV navigation based on deep reinforcement learning. In: 2019 IEEE International Conference on Signal, Information and Data Processing (ICSIDP). IEEE, pp. 1–4.
Zuo, G., Han, J., Han, G., 2010. Multi-robot formation control using reinforcement learning method. In: International Conference in Swarm Intelligence. Berlin, Heidelberg. Springer, pp. 667–674.