IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 22, 2019, accepted November 10, 2019, date of publication November 13, 2019,
date of current version November 25, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2953326

Learn to Navigate: Cooperative Path Planning for Unmanned Surface Vehicles Using Deep Reinforcement Learning

XINYUAN ZHOU[1], PENG WU[1], HAIFENG ZHANG[2], WEIHONG GUO[3], (Member, IEEE), AND YUANCHANG LIU[1]

[1]Department of Mechanical Engineering, University College London, London WC1E 7JE, U.K.
[2]Department of Computer Science, University College London, London WC1E 6BT, U.K.
[3]Department of Industrial and Systems Engineering, Rutgers University, New Brunswick, NJ 08854, USA

Corresponding author: Yuanchang Liu (yuanchang.liu@ucl.ac.uk)

ABSTRACT Unmanned surface vehicle (USV) has witnessed a rapid growth in the recent decade and has been applied in various practical applications in both military and civilian domains. USVs can either be deployed as a single unit or multiple vehicles in a fleet to conduct ocean missions. Central to the control of USV and USV formations, path planning is the key technology that ensures the navigation safety by generating collision free trajectories. Compared with conventional path planning algorithms, the deep reinforcement learning (RL) based planning algorithms provides a new resolution by integrating a high-level artificial intelligence. This work investigates the application of deep reinforcement learning algorithms for USV and USV formation path planning with specific focus on a reliable obstacle avoidance in constrained maritime environments. For single USV planning, with the primary aim being to calculate a shortest collision avoiding path, the designed RL path planning algorithm is able to solve other complex issues such as the compliance with vehicle motion constraints. The USV formation maintenance algorithm is capable of calculating suitable paths for the formation and retain the formation shape robustly or vary shapes where necessary, which is promising to assist with the navigation in environments with cluttered obstacles. The developed three sets of algorithms are validated and tested in computer-based simulations and practical maritime environments extracted from real harbour areas in the UK.

INDEX TERMS Deep reinforcement learning, motion planning, multi-agent systems, unmanned surface vehicles (USVs), USV formations.

I. INTRODUCTION

By witnessing the advance of technologies in robotics and autonomous systems (RAS) in recent decades, an growing interest has been cast on the development of unmanned surface vehicles (USVs) to support complex maritime missions. The deployment of USVs in both the civilian (such as environment monitoring, seabed mapping and research & rescue missions) and military (sea patrol and coastal guarding) domains has well demonstrated the benefits of using USVs including mitigated risks to personnel and increased mission efficiencies. However, due to the limited payload capacity and short endurance times, the capability of single USV is largely constrained making it necessary to seek alternative approach when conducting large-scale and complex ocean missions. As mentioned in [1] and [2], a promising approach to address these issues is to deploy multiple USVs as a formation fleet. By operating in a cooperative and collaborative manner, USV formations can provide appealing advantages such as wide mission area, improved system robustness and increased fault-tolerant resilience.

To support the operation of USV formations, a hierarchical structure has been proposed in [1] (shown in Fig. 1), where three different layers (Task management layer, Path planning layer and Task execution layer) are working interactively and seamlessly. Among these layers, the Path planning layer plays a critical role by proving key guidance information for formations. Optimised trajectories are generated by path planning algorithms within the Path planning layer to ensure USVs can effectively execute missions with guaranteed safety.

The associate editor coordinating the review of this manuscript and approving it for publication was Bohui Wang.

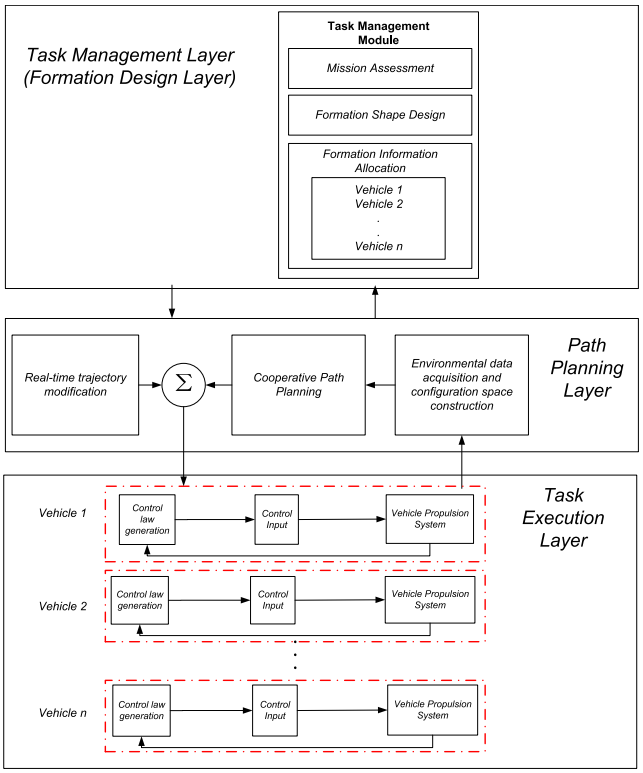


FIGURE 1. The hierarchical structure for USV formation [1].

Conventional path planning algorithms such as A* [3], artificial potential field [4] and fast marching method [5] have been successfully employed to calculate trajectories. However, it should be noted that all these methodologies adopt a repetitive strategy, where the environmental map needs to be continuously constructed and fed into the path planning algorithms to search for trajectories, hindering their adoption for real-time application. Also, when using conventional path planning algorithms for USV formations, such as in [1], the formation behaviour is achieved in a complex manner by using a leader-follower strategy, i.e, even though the final target remains unchanged for the leader USV, sub-targets have to be constantly calculated for follower USVs by adhering to complex mathematical relationships without taking the dynamic characteristics of USVs into account. Such a strategy makes the generated trajectories unsuitable for USVs to track in practical application, albeit some interesting features such as flexible formation shapes can be achieved.

In recent decades, the rapid advancement of artificial intelligence, especially reinforcement learning (RL), has opened new possibilities to solving path planning problems. As one of the three machine learning paradigms, the essence of RL is to train an agent to take an optimised action within an environment by maximising some notion of cumulative reward. Popular RL training methods include conventional training methods such as Q-learning, SARSA and artificial neural networks (ANNs) integrated training methods including Deep Q-Network (DQN), Deep Deterministic Policy Gradient (DDPG). Note that Q-learning trains an agent by maintaining a Q-table storing the training data and such a strategy falters with the increasing numbers of actions/states; whereas, ANN-based training methods such as DQN can be used in the scenario with high-dimensional state/action spaces. By adopting a trial-and-error training approach, differing to other two machine learning methods (supervised learning and unsupervised learning), RL does not require any human knowledge or pre-set rules, which enables it to achieve a high-level of human intelligence and become an appealing approach for USV path planning [6].

Such a human-level intelligence has been further amplified by several leading research undertaken by DeepMind over the last 5 years. For example, the concept of deep reinforcement learning (DRL) was first proposed in 'Playing Atari with Deep Reinforcement Learning' [7], where the combination of deep neural networks and reinforcement learning has been used to learn control strategies directly from high-dimensional input, i.e. an image data with raw pixels. Similarly, DRL was used to design and train a group of deep neural networks to plan the game of GO to compete with human players and achieve astonishing performances [8].

In terms of the application of reinforcement learning for USVs, the majority of research has been conducted on designing controllers for path tracking. Reference [9] designed a DRL based controller using deep deterministic policy gradient (DDPG) to achieve a self-learning capability to robustly follow a guidance trajectory. Reference [10] proposed an end-to-end DRL control strategy based upon Actor-Critic scheme to perform trajectory tracking in complex maritime environments. Ship's hydrodynamics as well as environmental influences such as wind and currents have been taken into consideration. A DRL based model identification method was proposed in [11]. By successfully capturing higher-order dynamic behaviours, the proposed deep learning algorithm is able to significantly reduce the motion prediction errors and greatly promote the robustness of the control of USVs. However, the application of DRL in the path planning for USVs has not been addressed.

In the field of reinforcement learning based path planning for USVs, limited work has been carried out. Reference [12] first proposed to use reinforcement learning (Q-learning) to generate an optimised trajectory for USVs in maritime environments subject to varying ocean currents and winds. The nonholonomic motion constraints of USVs have been considered and an additional path smoother was incorporated to improve the smoothness of trajectory. Reference [6] developed a knowledge-free path planning algorithm based upon Q-learning method for autonomous ships. By integrating the Nomoto ship model into Q-learning training process, a dynamics complaint trajectory can be generated in confined waterways. It should be noted that Q-learning based path planning algorithms can only be adopted in the case with finite (small) number of states, and its performance will be largely compromised in a continues state space, which is always the case for USV navigation. Reference [13] address

such an issue by employing DQN strategy for USV path planning with a special emphasis on obstacle avoidance.

By summarising these work it can be concluded that despite the high-level intelligence provided by reinforcement learning (RL), using RL to assist with path planning for USVs is still in its infancy and requires further investigation. Significant research gaps exist in the following aspects:

- RL based USV path planning needs to be carried out by addressing more practical constraints. At present, majority of work are based upon over-simplified assumptions in risk assessment and ship dynamic constraints, which largely hinders the deployment of algorithms on practical platforms.
- Initial studies have been undertaken to investigate USV formation path planning. However, these methods require a holistic navigation environment modelling and complex mathematical calculations for target point assignment. RL provides a new solution to formation path planning by working in a model-free manner that does not require sophisticated modelling. Presently, no RL based algorithms have been developed for USV formation path planning, which evidently needs more investigation.
- Most of the present studies only validate the algorithms in a simple 2D grid map with obstacles been modelled with regular shapes and the performance of RL based path planning in practical maritime environments needs to be investigated.

Central to the RL problem are agents, environments and their interactions, which can be mathematically modelled by Markov decision process (MDP) (details of MDP will be described in the next section) and it can be argued that the success of RL training is largely underpinned by a proper construction of MDP. Therefore, in this paper, to resolve the above-mentioned issues, new deep reinforcement learning based path planning algorithms have been proposed and designed for single USV and USV formations applications. New MDPs are constructed for single USV and USV formation scenarios and the DQN algorithm is used for RL training. Specifically, for USV formation path planning, crucial formation features such as formation maintenance and formation shape variation have been considered and achieved by designing new reward functions within the USV formation MDP. The designed algorithms have not only been tested in self-constructed simulation environments but practical maritime environments extracted from real harbour areas in the UK.

The rest of the paper is organised as follows. Section 2 specifically introduces fundamentals in reinforcement learning including MDP and key training algorithms such as Q-learning. In Section 3, the detailed algorithms for single USV path planning and USV formation path planning is introduced, which includes the newly constructed MDPs and the DQN algorithm. The proposed algorithms and methods are verified by simulations in Section 4. Section 5 concludes the paper and discusses the future work.

II. FUNDAMENTALS IN REINFORCEMENT LEARNING

In this section, the rationale of reinforcement learning (RL) will be discussed. The Markov Decision Process (MDP) which is normally used for reinforcement environment modelling will be first introduced and then followed by the discussion on one of the fundamental RL algorithms - Q-learning algorithm.

A. MARKOV DECISION PROCESS

A Markov Decision Process (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:

$$M = [s, a, p, r] \tag{1}$$

where $s = s_1, s_2, ..., s_t, s_{t+1}$ represents the dynamic environment with a finite set of states with s_t denoting the state at time t. $a = a_1, a_2, ..., a_t, a_{t+1}$ 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:

$$p^a_{ss'} = \mathbb{P}\left[s_{t+1} = s' | s_t = s, a_t = a\right] \tag{2}$$

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 r_{t+1} to the agent [14].

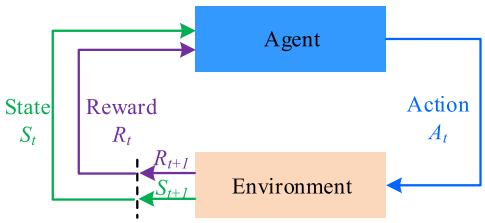


FIGURE 2. The agent-environment MDP interaction framework.

B. Q-LEARNING

Q-learning (pesudocode shown in Algorithm 1), proposed by [15], is one of the early breakthroughs in reinforcement learning [14]. The aim of Q-learning is to find an optimal control policy π^* for a given MDP. π^* maximises the action-value function:

$$Q^*(s, a) = \max_{\pi} \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, a_t = a, \pi\right] \tag{3}$$

Q-learning is an off-policy approach solving MDPs, and directly approximates the action-value function via:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a} Q(s', a) - Q(s, a)] \tag{4}$$

Algorithm 1 Q-Learning Pseudo Code

```
Initialise Q(s, a) arbitrarily except that
Q(terminal, ·) = 0
foreach episode do
    Initialise initial state s
    foreach step of episode do
        Choose a with s using policy (e.g. ε-greedy)
        derived from Q
        Take action a, observe r, s'
        Q(s, a) ←
        Q(s, a) + α[r + γ max_a Q(s', a) − Q(s, a)]
        s ← s'
        Break if s is terminal
    end foreach
end foreach
```

where $\alpha \in [0, 1]$ is the learning rate. Note that Q-learning algorithm needs to initialise a Q table to record the expectation of action-value function making it only suitable for dealing with simple situations with small action and state space such as maze problems. However, majority of real-world problems need a huge number of states and action spaces, making it impossible to build a Q table to contain all the states and actions.

III. DEEP REINFORCEMENT LEARNING BASED COOPERATIVE MOTION PLANNING FOR USVS AND USV FORMATIONS

In this section the designed MDPs for single USV formation USV formation path planning will be described after the introduction of the kinematic motion of USVs which predominately regulates the state transition within the MDP. Also, the deep Q network will be explained and shows how the defined MDPs are trained in a deep learning manner.

A. THE KINEMATIC MOTION OF USVS

The basic kinematic motion equations of USVs will be first explained. Consider $\langle e \rangle$ is the inertial coordinate frame and $\langle b \rangle$ is the body fixed coordinate frame. Let the state of the USV relative to $\langle e \rangle$ be denoted as $\boldsymbol{\eta} = \begin{bmatrix} x & y & \phi \end{bmatrix}^T$, where x and y represent the position coordinates of the USV in the planning space and ϕ is the heading direction. The surge and yaw motion of the USV is expressed with respect to $\langle b \rangle$ and has the form of $\boldsymbol{v} = \begin{bmatrix} u & v & w \end{bmatrix}^T$, where u and v are surge and sway motion and w is the yaw rate. The kinematic motions of the USV can therefore be written as:

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}\boldsymbol{v} \tag{5}$$

where

$$\boldsymbol{J} = \boldsymbol{J}(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

The kinematic motion model of USV will be filly incorporated into the designed Markov decision processes (MDPs) for USV path planning.

B. MDP FOR SINGLE USV PATH PLANNING

1) STATE SPACE

For the MDP for single USV path planning, the state space is defined to include USV's position coordinate (x, y) and heading direction θ. As shown in Fig. 3, where the reinforcement learning (RL) training is carried out in an environment having the dimension of 500×500 pixels, any state with the position coordinate exceeding the range of [0, 500] will be regarded as invalid and any movement into such a state will be penalised.

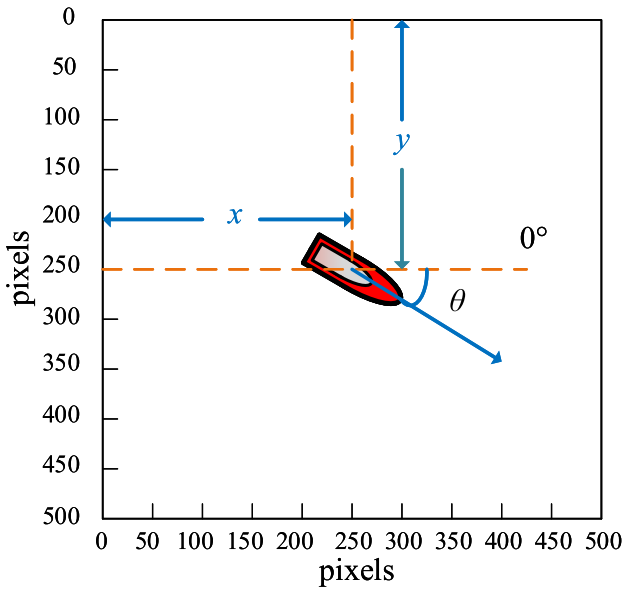


FIGURE 3. Illustration of the state space of a single USV in Cartesian coordinate frame.

2) ACTION SPACE

By considering the motion characteristics of USVs, in this paper, it is assumed USVs will maintain its surge speed during the navigation and actions taken by USVs are the angular velocities which will lead to changes in vehicle's heading directions. Different from most of existing work in using RL for USV path planning, where the action space is defined to be consisting of 4 discrete actions, i.e., *up*, *down*, *left*, *right* ([16], [17]), a new action space with refined discrete actions is defined in this work as:

$$A = \begin{bmatrix} a_1 & a_2 & \dots & a_7 & \dots & a_{12} & a_{13} \end{bmatrix} \tag{7}$$

where the action space A is a vector containing 13 elements with the value ranging from $-60°(a_1)$ to $60°(a_{13})$ and with increments of 10°. Such a design is largely complaint with the dynamics characteristics of USVs, i.e. a USV can only make a turning within certain range during one control episode [5]. It should be noted that the range of $[-60°, 60°]$ is set up based upon our previous experiments conducted on a practical USV platform, *Springer* USV ([18]) and this value can be changed when the algorithms are used on other platforms. According to the defined state and action spaces, the state transition with a given action a_i by adhering to the kinematics of a USV can

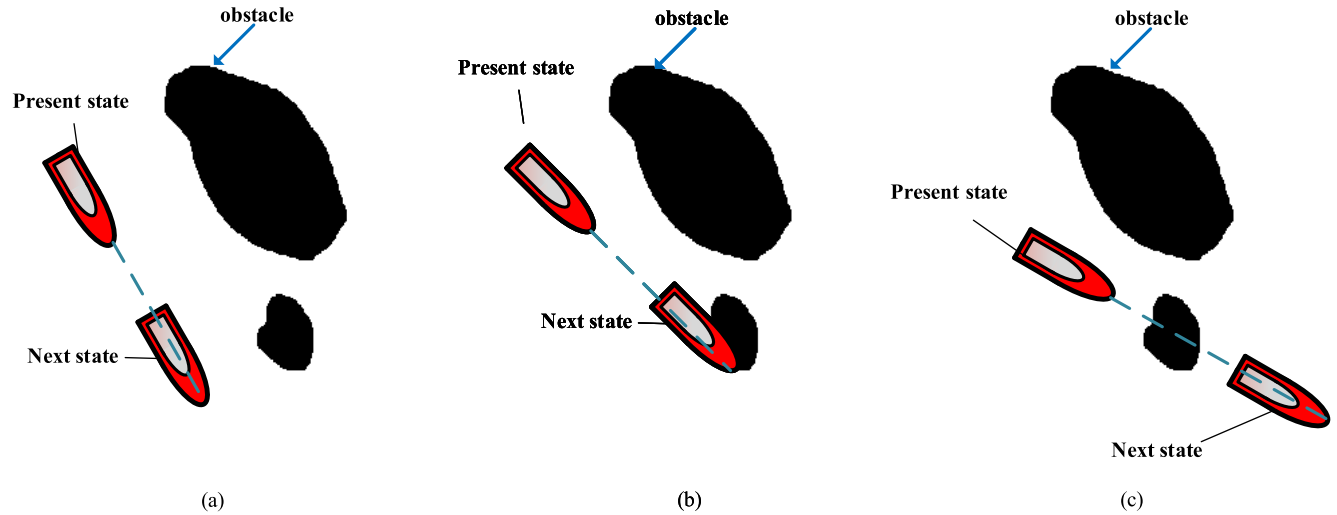


FIGURE 4. The schematic diagram for collision avoiding reward function. (a) A situation when no collision happens. (b) A collision happens in the next state. (c) A collision happens along the route.

be calculated as:

$$\begin{aligned} x' &= x + x\cos\phi' \\ y' &= y + y\sin\phi' \\ \phi' &= \phi + a_i \end{aligned} \tag{8}$$

where $\begin{bmatrix} x' & y' & \phi' \end{bmatrix}$ is the state in next step, $\begin{bmatrix} x & y & \phi \end{bmatrix}$ is the state in current step and a_i is the chosen action in current step.

3) REWARD FUNCTIONS

A reward function in reinforcement learning (RL) should be designed to enforce an agent to learn a desired behaviour and complete defined tasks as expected. For single USV navigation, in order to generate the shortest path reaching goal areas as well as prevent any collisions *en route*, a set of sub-reward functions have been defined in this work and the total reward returned by the environment is the sum of all sub-rewards.

a: TERMINAL REWARD FUNCTION ($r_{terminal}$)

The terminal reward is designed to encourage USVs to reach the goal area and will only be activated when a USV manages to reach this area. A large positive value should be applied to $r_{terminal}$ as the task of reaching the goal area is one of the most critical criteria of completing a mission.

b: COLLISION AVOIDING REWARD FUNCTION ($r_{obstacle}$)

The collision avoiding reward function is designed to prevent any collisions. During the RL training process, once a collision occurs between a USV and an obstacle, the collision avoiding function will return a large negative reward to 'punish' such an unfavoured action in current state and 'suggest' the vessel not to take this action if similar scenario occurs in future training stages. The collision avoiding reward function has a strong correlation with the state of vessel and $r_{obstacle}$ can be considered as assigning negative rewards for some specific actions in following categories:

- *Normal Condition:* If no collision happens for the next state (Fig. 4 (a)), no negative reward will be triggered.
- *Collision in Next State:* If it is detected that by taking the selected action a collision would happen in the next state (Fig. 4 (b)), a large negative reward $r_{obstacle}$ will be assigned in this condition to discourage taking current action at such a state.
- *Collision Along the Route:* In some cases, a collision will not happen in the next state of vessel (the vessel's coordinate will not fall into the obstacle area) but along the trajectory during the transition from current state to the next state (Fig. 4 (c)). Such a situation should be well cleared. The path by taking each action will be checked to avoid such a condition happening. If an obstacle appears in the path, a large negative reward $r_{obstacle}$ will be returned as well.

c: DIRECTION CHANGING REWARD FUNCTION (r_ϕ)

it aims to ensure the adjustment of vessels' headings in a desired manner with two underlying sub-tasks defined as: 1) a reward ($r_{\Delta\phi}$) is designed to constrain the heading change incurred by taking each action within a small range such that a smooth heading transition can be achieved and 2) a reward (r_{ϕ_0}) is designed to encourages USVs arriving at the destination area with an ideal reference direction (ϕ_0). The specific reward functions are as follows:

$$\begin{aligned} r_\phi &= r_{\Delta\phi} + r_{\phi_0} \\ r_{\Delta\phi} &= -\lambda_{\Delta\phi}|\phi' - \phi| \\ r_{\phi_0} &= -\lambda_{\phi_0}|\phi' - \phi_0| \end{aligned} \tag{9}$$

where ϕ' and ϕ are heading directions in next and current states, respectively. ϕ_0 is the desired heading for USVs to

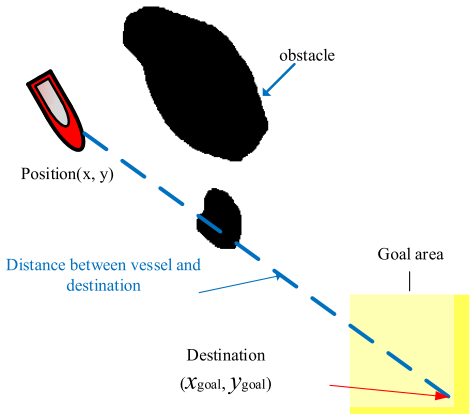


FIGURE 5. The schematic diagram of distance reward function.

reach the destination. $\lambda_{\phi_0} > 0$ and $\lambda_{\Delta\phi} > 0$ are the direction reward coefficients for r_{ϕ_0} and $r_{\Delta\phi}$ respectively.

d: DISTANCE REWARD FUNCTION ($r_{distance}$)

it trains USVs to find the shortest path to destination. Since such a task is one of the most important components for path planning, in order to enhance the influence of destination, the distance reward function has been designed in a way that the closer the distance between a USV and the destination, the less punishment would be imposed. The distance reward function can be expressed as:

$$r_{distance} = -\lambda_{distance}\sqrt{\left(x - x_{goal}\right)^2 + \left(y - y_{goal}\right)^2} \quad (10)$$

where x_{goal} and y_{goal} represent the coordinate of the target point; $\lambda_{distance}$ is a constant which defines the importance of this reward. Note that differing to most of existing work of RL based USV path planning ([6] and [16]), to increase the training efficiency, arriving at a surrounding area of the goal point (the yellow square area in Fig. 5) instead of the exact location is regarded as accomplishing the mission.

e: SURROUNDING OBSTACLE REWARD FUNCTION (r_{sur})

This reward has been designed to provide an enhanced safety for USV navigation. In a typical littoral environment, not only obstacles but the shallow water areas surrounding obstacles are not permissible for travelling and USVs should keep a well distance away to these areas. Therefore, to resolve such an issue, a safety area has been constructed for a USV and any movement that will lead to the violation to safety areas will be penalised. The safety area in this work is designed to be a square area with the USV been centred and the length of area being 21 pixels. If an obstacle occurs in the safety area, a punishment will be assigned to the agent. As shown in Fig. 6 (a), when there is no obstacle appearing in the area (represented as the region of interest (ROI)), the surrounding obstacle reward (r_{sur}) would be assigned as 0; whereas, in Fig. 6 (b) although collisions would not happen in the next state, the obstacle occurs in the safety area which would cause a potential threat, hence a negative reward r_{sur} will be returned as a punishment. The value of r_{sur} will be configured according to specific application scenarios.

C. MDP FOR COOPERATIVE USV FORMATION PATH PLANNING

1) STATE SPACE

In this work, a typical formation configuration consisting of three USVs is considered as the research object and two critical formation problems, i.e. the formation maintenance and formation variations, have been properly investigated. To facilitate the control of USV formations, the leader-follower formation control strategy has been adopted with one USV been assigned as the leader USV and the other two as the followers. A centralised coordination scheme is used for training the USV formations and a state space for 3-USV formation system consists of 9 features including the position (x, y) and heading direction ϕ for each vessel (Fig. 7) can be defined as:

$$S = \begin{bmatrix} x_1 & y_1 & \phi_1 \\ x_2 & y_2 & \phi_2 \\ x_3 & y_3 & \phi_3 \end{bmatrix} \quad (11)$$

where $s_1 = \begin{bmatrix} x_1 & y_1 & \phi_1 \end{bmatrix}$ is the leader USV's state, and $s_2 = \begin{bmatrix} x_2 & y_2 & \phi_2 \end{bmatrix}$ and $s_3 = \begin{bmatrix} x_3 & y_3 & \phi_3 \end{bmatrix}$ are the states of two follower USVs.

2) ACTION SPACE

It is assumed that each USV can take three possible actions from an action space $a_i \in [-10°, 0°, 10°]$. Different from single USV path planning, USV formation algorithm controls three vessels simultaneously and the resulting next state of the formation system can be calculated as:

$$s' = \begin{bmatrix} x_1\cos(\phi_1 + a_1) & y_1\sin(\phi_1 + a_1) & \phi_1 + a_1 \\ x_2\cos(\phi_2 + a_2) & y_2\sin(\phi_2 + a_2) & \phi_2 + a_2 \\ x_3\cos(\phi_3 + a_3) & y_3\sin(\phi_3 + a_3) & \phi_3 + a_3 \end{bmatrix} \quad (12)$$

3) REWARD FUNCTIONS

As an extension of the single USV path planning problem, the majority of reward functions for formation systems are similar to single planning case excepting the rewards designed for formation behaviours, i.e. formation maintenance and variations. Specific rewards are illustrated as:

Terminal reward function ($r_{terminal}$): when the leader USV reaches the destination area, current training episode terminates and the environment will returns a positive reward $r_{terminal}$ to the agent.

Collision avoiding reward function ($r_{obstacle}$): similar to single USV path planning scenario, this function aims to impose a large negative reward to any action leading to potential collisions. Differences are for formation algorithm the collision avoiding reward is be superimposed, i.e. if one vessel hits the obstacle, a large negative punishment $r_{obstacle}$ would be assigned; whereas, if collisions are detected for two vessels in the same time step, a punishment of $2r_{obstacle}$ would be assigned.

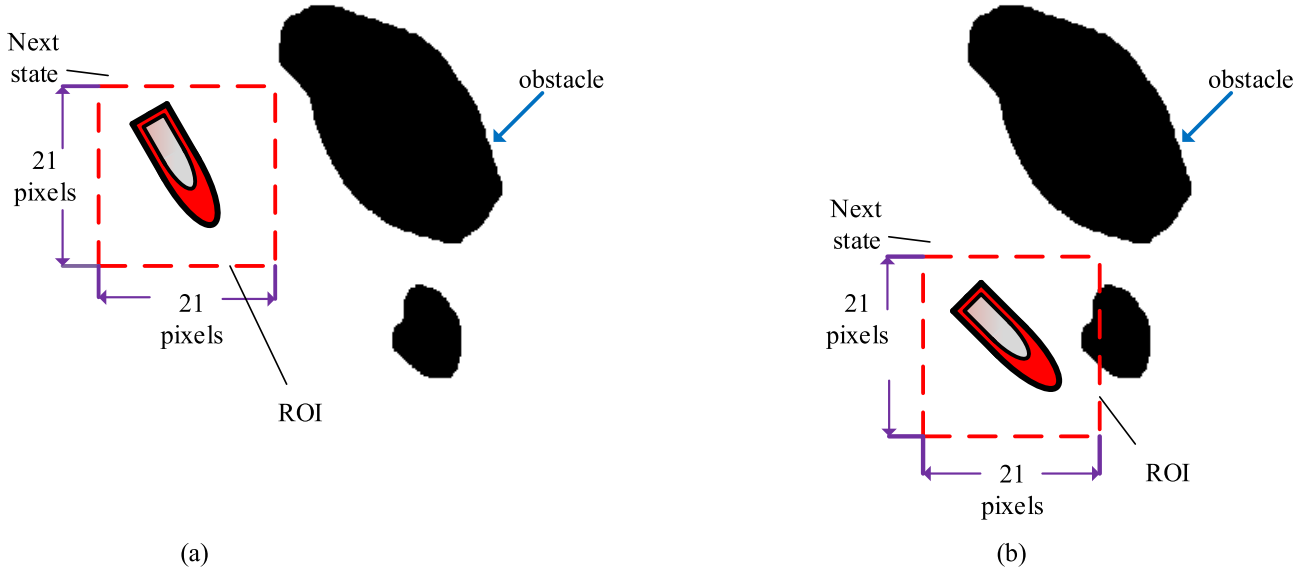


FIGURE 6. The schematic diagram of surrounding reward function. (a) A condition when no obstacle appears in the safety area. (b) A condition when obstacle appears in the safety area.

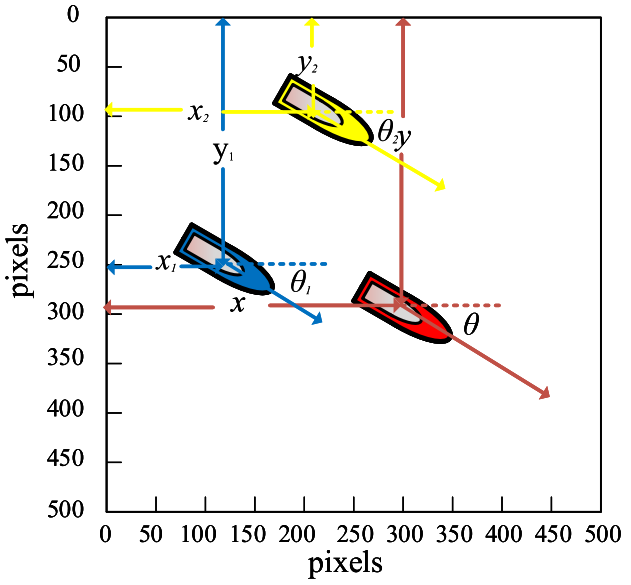


FIGURE 7. Illustration of the state space of a USV formation in Cartesian coordinate frame.

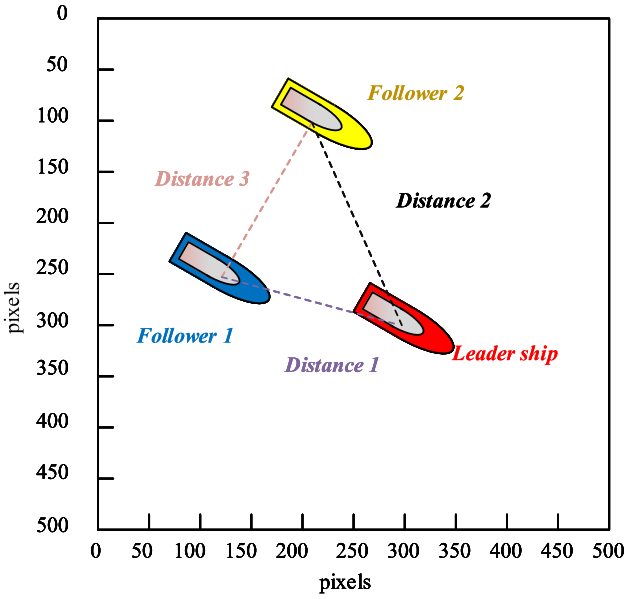


FIGURE 8. Schematic of formation distance reward function.

Distance reward function ($r_{3distance}$): for USV formations, as USVs are travelling as a group with relative close distances to each other, it is regarded as successfully reaching at the target point as long as the leader USV arrives. Therefore, the distance reward function is designed to encourage the leader USV to find the shortest path to destination and the reward function can be expressed as:

$$r_{3distance} = -\lambda_{3distance}\left(|x_1 - x_{goal}| + |y_1 - y_{goal}|\right) \quad (13)$$

where $\lambda_{3distance}$ is a constant which aims to define the importance of this reward and (x_1, y_1) is the position of leader USV and (x_{goal}, y_{goal}) is the coordinate of goal point.

Formation distance reward function (r_{fd}): The formation shape maintenance is one of essential tasks and in this work a triangular shape is selected as the desired formation to be retained. Note that other shapes can also be achieved by adhering to the proposed algorithm in this paper. Based upon the leader-follower structure, i.e. followers need to keep a desired distance to the leader so as to maintain the formation shape, a schematic of formation distance reward function to control the distances between pair of USVs as illustrated in Fig. 8. The range for *distance 1* and *distance 2* denote the distances between the leader USV and follower 1 and follower 1, respectively; whereas, *distance 3* denotes

the distance between two followers. The specific formation distance reward is calculated as:

During each training step, distances between each pair of USVs are calculated and compared against if all the distances are within the range, no punishment would be given to the agent. However, if a distance between two vessels is out of range, a negative reward r_{fc} would be assigned to the agent as a punishment. The formation distance reward is stackable:

$$\begin{aligned} r_{fd} &= r_{dis1} + r_{dis2} + r_{dis3} \\ r_{dis1} &= \begin{cases} 0 & \sqrt{(x-x_1)^2 + (y-y_1)^2} \in D_1 \\ r_{fd} & else \end{cases} \\ r_{dis2} &= \begin{cases} 0 & \sqrt{(x-x_2)^2 + (y-y_2)^2} \in D_2 \\ r_{fd} & else \end{cases} \\ r_{dis3} &= \begin{cases} 0 & \sqrt{(x_2-x_1)^2 + (y_2-y_1)^2} \in D_3 \\ r_{fd} & else \end{cases} \end{aligned} \tag{14}$$

where r_{dis1}, r_{dis2} and r_{dis3} represents values of reward function for *distance 1*, *distance 2* and *distance 3*, respectively; (x, y); (x_1, y_1); and (x_2, y_2) are position states of leading ship, follower 1 and follower 2 respectively; d_1, d_2 and d_3 are the desired ranges for *Distance 1*, *Distance 2* and *Distance 3*, respectively. Note that to facilitate the adoption of a flexible formation shape, d_1, d_2 and d_3 has been configured with the relationship of $d_1 = d_2 < d_3$ so that two followers are allowed to travel with a slightly larger distance. r_{fd} is the negative reward assigned to an USV if the calculated relative distance is out of range.

Formation position reward function (r_{fp}): it aims to adjust the relative positions of three USVs while maintaining the triangular formation. In most instances, the sole control of distances between each USV is effective enough to maintain the shape. For example, two followers may swap their positions during voyage without any further constraints which could an internal collision. Similarly, two follower USVs can possibly overtake the leader USV which does not satisfy the underlying requirement of leader-follower structure. To solve these issues, it is important to consider the relative positions between each USV and the formation position reward function has therefore been contrived by taking three criterion into consideration. Considering the general structure of the formation shown in Fig. 8, it is first assumed that follower 2 should stay above the leader as well as follower 1 in the coordinate frame. In addition, the leader USV should be on the right side of follower 1. The reward r_{fp} is defined in a stackable way and can be expressed as:

$$\begin{aligned} r_{fp} &= r_{pos1} + r_{pos2} + r_{pos3} \\ r_{pos1} &= \begin{cases} 0 & y_1 < y \\ r_{fp} & else \end{cases} \\ r_{pos2} &= \begin{cases} 0 & y_1 < y_2 \\ r_{fp} & else \end{cases} \\ r_{pos2} &= \begin{cases} 0 & x_1 < x \\ r_{fp} & else \end{cases} \end{aligned} \tag{15}$$

where r_{pos1}, r_{pos2} and r_{pos3} denote individual reward function for three qualifications, respectively; r_{fp} is the negative reward assigned to the agent.

Action reward function (r_{action}): this reward has been proposed to encourage USVs within the formation to choose unified and coordinated actions to reinforce the formation shape maintenance. If the taken action would lead to a different heading direction, a negative reward (r_{action}) will be applied to the agent as a punishment. Note that such a reward function will not be employed when a flexible formation shape is considered.

To summarise the proposed reward functions for USV formation path planning, two difference cases have been specifically considered:

- *Formation maintenance*: six rewards ($r_{terminal}$, $r_{obstacle}$, $r_{3distance}$, r_{fd}, r_{action} and r_{fp}) have been designed. With $r_{terminal}$ being the only reward having a positive value, the rest rewards have different negative values according to the their importance for a path planning task and the absolute values of these rewards have the relationship of:

$$|r_{obstacle}| > |r_{action}| > |r_{fd}| > |r_{fp}| > |r_{3distance}| \tag{16}$$

- *Formation variations*: five rewards ($r_{terminal}$, $r_{obstacle}$, $r_{3distance}$, r_{fd} and r_{fp}) will be used and the absolute values of these rewards is ranked as:

$$|r_{obstacle}| > |r_{fd}| > |r_{fp}| > |r_{3distance}| \tag{17}$$

D. DEEP Q NETWORK

The designed USV single and formation path planning MDPs are trained using deep Q Network (DQN) [19]. Similar to Q-learning, DQN (Algorithm 2) is also an off-policy algorithm. As shown in Fig. 9, the main components of DQN algorithm are the environment, 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 [19]. 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 the 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

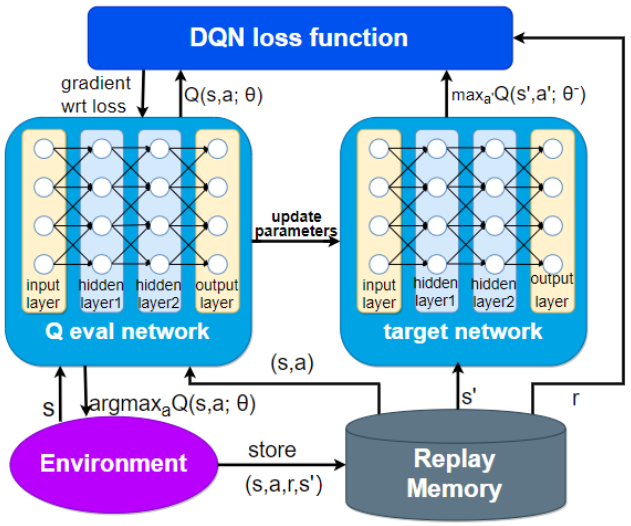


FIGURE 9. Schematic of Deep Q network.

θ^- update periodically. Such an update strategy improves the training stability.

At each training step, the target Q network outputs an approximate target action value using parameters θ^- from some previous iterations, with given states:

$$z_i = r_i + \gamma \max_{a'} Q(s', a', \theta^-) \tag{18}$$

The loss function can be considered as the difference between the true action value and target action value which is expressed as:

$$L_i(\theta_i) = \mathbb{E}_{s,a,r}\left[(z_i - Q(s, a; \theta))^2\right] \tag{19}$$

The aim of training is to minimize the loss function according to the following gradient with respect to the parameters θ_i of the Q-network at iteration i:

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a,r}\left[(z_i - Q(s, a; \theta_i))\, \nabla_{\theta_i} Q(s, a; \theta_i)\right] \tag{20}$$

The loss function can be optimised via stochastic gradient descents by sampling mini-batches with capacity of N from the replay memory D. Note that this approach is off-policy, i.e. the agent acts greedily following ε-greedy policies when choosing next actions:

$$a_t = \begin{cases} \arg\max_a Q^{\pi}(s_t, a), & \text{with probability } \varepsilon \\ \text{random action } a_t, & \text{otherwise} \end{cases} \tag{21}$$

With action a_t taken by the agent, the environment returns reward r_t and next system state s_{t+1} as described in Fig. 9. Such a transition (s_t, a_t, r_t, s_{t+1}) will then be stored into the replay memory D. Note that new transitions override old experiences when the memory is full.

Algorithm 2 Deep Q-Network Pseudo Code

Initialise the memory replay D to with capacity N
Initialise the Q network with random parameters θ
Initialise the $\hat{Q}$ target network with parameters $\theta^- = \theta$
Set maximum step number $step_{max}$
foreach *episode* **do**
  Initialise initial state s
  foreach *step of episode* **do**
    With probability ε choose random action a_t
    Otherwise select $a_t = \arg\max_a Q(s, a; \theta)$
    Execute a_t, observe the reward r_t and the next state s_{t+1}
    Store (s_t, a_t, r_t, s_{t+1}) into replay memory D
    Sample random mini-batch of transition (s_j, a_j, r_j, s_{j+1}) from D
    Set
    $$z_j = \begin{cases} r_j, \text{ if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}\left(s_{j+1}, a'; \theta^-\right), \text{otherwise} \end{cases}$$
    Update θ via a gradient descent step on loss function $\left(z_j - Q\left(s_j, a_j; \theta\right)\right)^2$
    Every C steps $\hat{Q} \leftarrow Q$
  end foreach
end foreach

IV. SIMULATION RESULTS AND DISCUSSIONS

This section provides detailed analyse and discussions on the simulation results of the proposed algorithms. Simulations are conducted to validate the capability of algorithms in three aspects: 1) single USV path planning in dealing with collision avoidance, 2) USV formation path planning with the emphasis on maintaining a formation shape, 3) USV formation path planning with the emphasis on formation variations. All simulations are carried in a 2D binary map with the dimension of 500 pixels × 500 pixels. Furthermore, two practical maritime environments near Portsmouth harbour and Plymouth harbour, UK, are adopted to verify the algorithms' performance in dealing with practical navigation issues.

The algorithms are coded in Python with the deep neural networks built and trained using TensorFlow v1.14. For all the experiments carried out in this study, the Q networks are designed with two fully connected hidden layers. The input layer and each hidden layer are applied with Rectified Linear Unit (ReLU) activation functions. The Q networks are trained with the RMSProp optimiser [20]. Simulations are run on a workstation with Intel Xeon E5-2683 V3 2.4 GHz processor. To provide a comprehensive evaluation and discussion on the simulation results, specific training settings together with training processes will be first introduced, followed by the discussions on the generated trajectories using proposed deep reinforcement learning algorithms.

The DQN agent training process requires careful balancing between exploration and exploitation. With a suitable level of exploration, the agent can gain adequate experience which would provide information to make the best decisions with given states by exploitation. In the USV path planning problem, the exploration is driven by the ε-greedy policy.

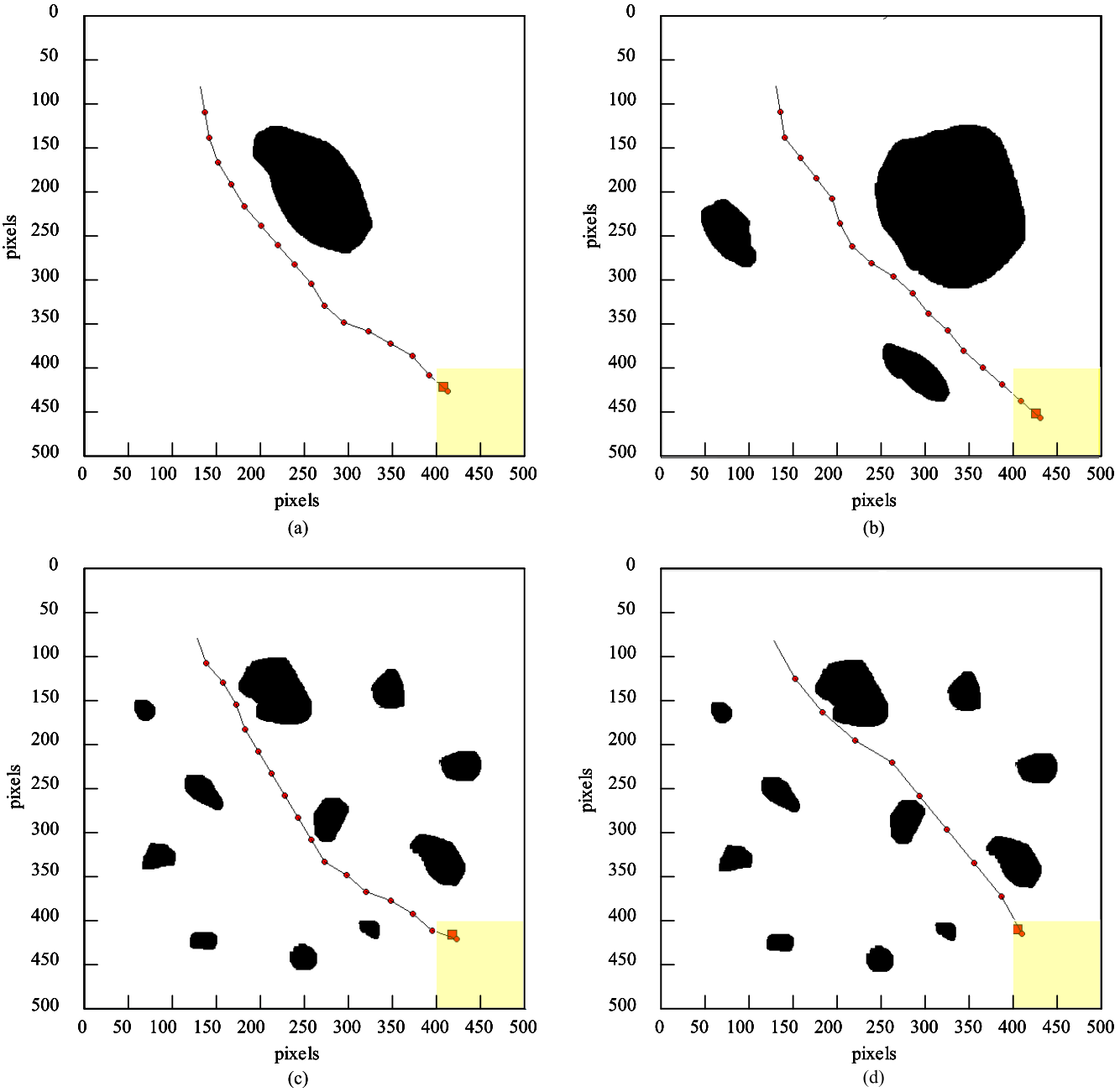


FIGURE 10. The simulation results for USV motion path planning algorithm. (a) A simple navigation map with the velocity of 30 pixels per step. (b) A map with intermediate difficulty with the velocity of 30 pixels per step. (c) A map with complex environment with the velocity of 30 pixels per step. (d) A map with complex environment with the velocity of 50 pixels per step.

At each training step, with probability ε, the agent takes a random maneuvering action from the action space; otherwise an action with maximum action-value function will be taken. The ε of the ε-greedy policy is fixed at 0.1 in all experiments. The learning rates are fine tuned for different simulations scenarios.

A. SINGLE USV PATH PLANNING

1) TRAINING SETTINGS

In this section, two speed magnitudes, i.e., 30 and 50 pixels per time step, have been simulated to test the algorithm's capability in collision avoidance at different speeds. Table 1 depicts the hyper parameter and neural network (NN) settings for the single USV path planning algorithm. For the speed of 30 pixels per step, it would require more time steps to reach the target area; the agent would inevitably encounter more intermediate states. Therefore, the Q-network's first hidden layer neurons have been increased from 8 to 12 to handle the increased input complexity. However, after trial and error, a larger learning rate of 0.03 is chosen for the lower speed scenario, while it is 0.007 for the speed of 50 pixels per step. A lower speed means the USV can be controlled relatively

TABLE 1. Agent training settings for single USV path planning.

Parameter	30-pixel speed	50-pixel speed
Learning rate	0.03	0.007
Discount rate	0.9	0.9
ε	0.1	0.1
Memory size	2×10^4	2×10^4
Mini-batch size	64	64
$\hat{Q}$ update frequency	3×10^3 steps	3×10^3 steps
NN hidden layer 1	12 neurons	8 neurons
NN hidden layer 2	13 neurons	13 neurons

easier to avoid collisions; therefore, a larger learning rate can be applied to gain useful experience. Other training parameter settings are listed in Table 1 as well. All simulations were terminated after 4×10^5 episodes of training.

2) RESULTS AND ANALYSIS

The trained deep Q network outputs heading direction control actions by observing the USV state inputs. Fig. 10 presents the trajectories of a USV controlled by the DQN polices. Note that the speed magnitude is 30 pixels for Fig. 10 (a)-(c) and 50 pixels for Fig. 10 (d), respectively. In Fig. 10, obstacles are represented by irregular areas in black and the black lines display the optimal trajectory calculated by the algorithm. Red dots represent the locations of USV after each action has been executed with the goal point in a red square. A large yellow squared area in the bottom right part of the simulation environment demonstrates the dimension of a destination area, and any movement steps into this area will be regarded as successfully reaching the goal point.

In Fig. 10 (a)-(b), relatively simple simulation environments have been adopted with one and three obstacles randomly incorporated. It can be observed that trajectories with short distance and well clearance to obstacles can be generated in both cases. More importantly, although according to the defined action space the heading of USV can change from $-60°$ to $60°$, the direction changing reward function (r_ϕ) is able to constrain the heading changes within a small margin which effectively increases the smoothness of trajectories. When the trained deep Q network is applied in a more complex environment (shown in Fig. 10 (c)), a collision free trajectory can also be successfully generated and well reaches the goal point. Within the same environment, when the speed of USV is increased to 50 pixels, a shorter trajectory (shown in Fig. 10 (d)) will be produced by the deep Q network. Such a decrease in total distance is achieved upon the sacrifice of safety where the trajectory in Fig. 10 (d) stays closer to certain obstacles. Another advantage of using larger speed is that the training time can be effectively reduced because less time will be spent on completing one training episode. Therefore, it can be recommended that large speed is suitable for generating a global trajectory, whereas small speed is better for local planning.

In terms of the specific training process, Fig. 11 shows the moving average episode reward with a moving windows size of 1000 episodes for the scenario presented in Fig. 10(c). Note that similar reward trends have been observed for the other three maps for single USV path planning. The DQN agent initially learned a policy leading to a mean episode reward of -132 (around 0.4×10^4 episode), suggesting the agent has gained negative experiences to a maximum degree. From this point onward, the agent managed to improve the USV control policy gradually, and the mean episode reward converged to approximately -10 after about 3×10^4 episodes of training. The training was terminated at 4×10^4 episodes. Similar training processes have been observed in the other three experiments carried out in this section.

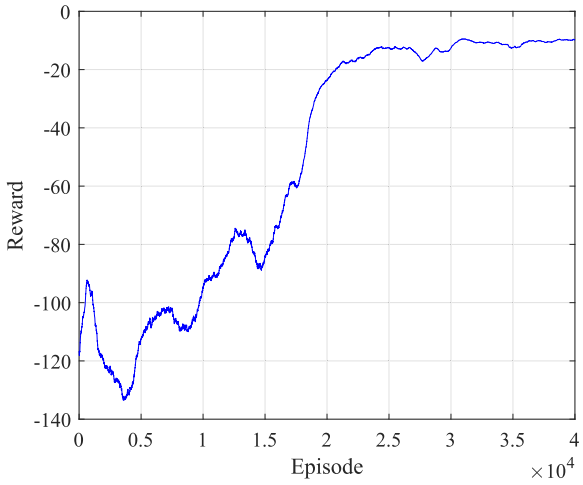


FIGURE 11. Single USV path planning training process of Fig. 10 (c): moving average episode reward with a moving window size of 1000 episodes.

B. USV FORMATION PATH PLANNING WITH FORMATION SHAPE MAINTENANCE

Simulations in this section aims to validate the algorithm's capability of planning trajectories for USV formations whilst maintaining a predefined formation shape. In general, a formation shape can be well retained if the same actions can be taken by each USV at every time step. Therefore, to fully test the algorithm's performances, various simulation environments with varying number of obstacles have been used. Note that to simplify the training process, the same start and goal points are used in all cases.

TABLE 2. Agent training parameters for USV fixed shape formation.

Parameter	Value
Speed	50 pixels per step
Learning rate	0.007
Discount rate	0.9
ε	0.1
Memory size	2×10^5
Mini-batch size	64
$\hat{Q}$ update frequency	3×10^3 steps
NN hidden layer 1	20 neurons
NN hidden layer 2	27 neurons

1) TRAINING SETTINGS

Table 2 details the agent training settings for the USV formation path planning with fixed shapes. The USV speed is set at

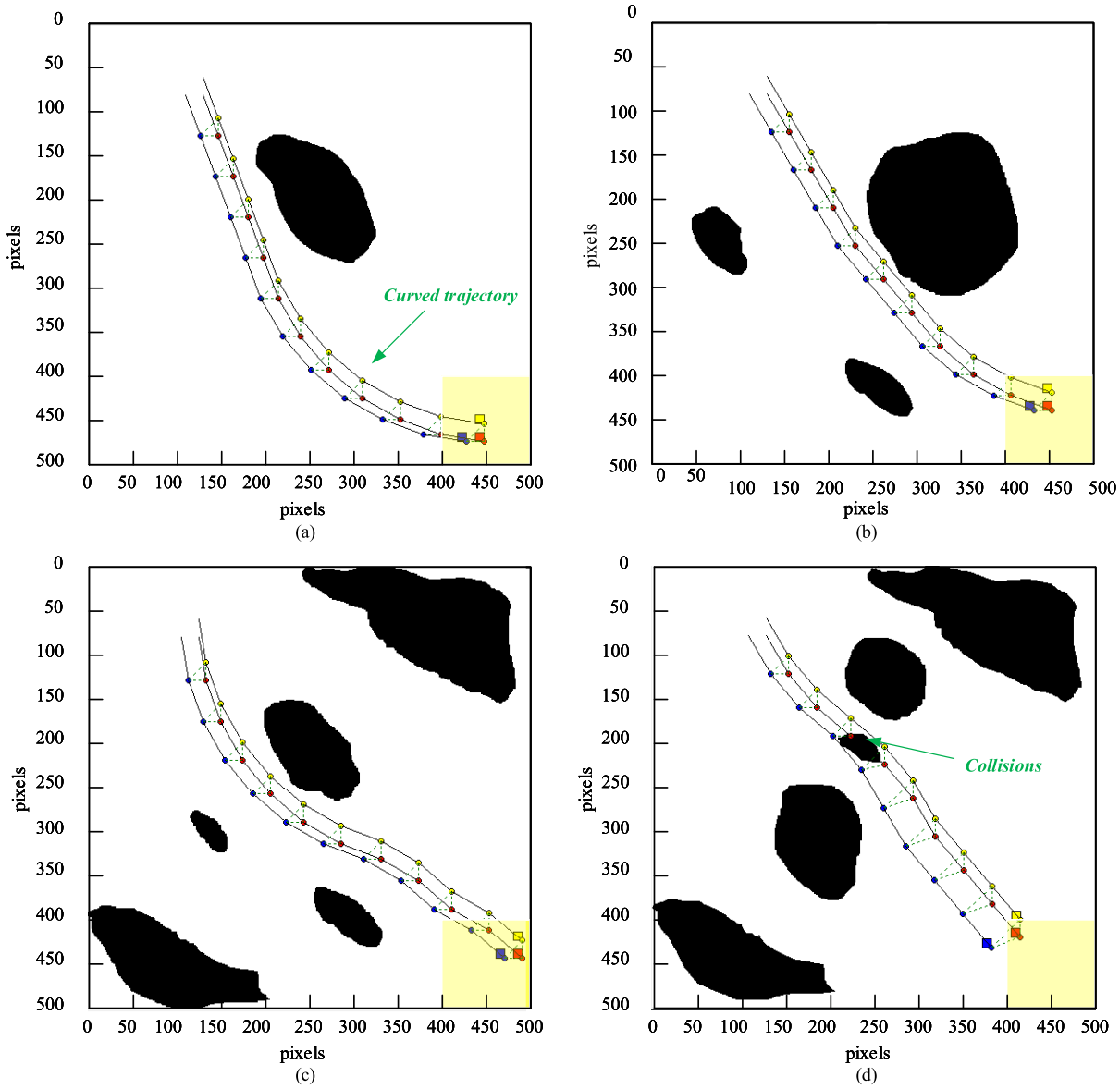


FIGURE 12. The simulation results for USV formation path planning algorithm with formation shape maintenance capability. (a) Simulation results in simple environment with one obstacle. (b) Simulation results in three obstacles environment. (c) Simulation results in relative complex environment with five obstacles. (d) Failed simulation results in difficult environment with dense distribution of obstacles.

50 pixels per time step to investigate the algorithm's capability of maintaining a fixed shape at high speed. As the input features to the Q-network is increased from 3 to 9 (3 USVs, each has 3 state features), the numbers of neurons in each layer are increased to 20 (hidden layer 1) and 27 (hidden layer 2) respectively. It is worth noting that the training episode number has increased from 4×10^4 to 1×10^5 due to the increased state dimension and number of actions. Such an increased training episode number ensures that the agents can find reasonable policies. The replay memory has been increased to 200 000. The Q-target network was trained with RMSProp optimiser, and the learning rate was fixed at 0.007 for all the fixed shape formation training. Other important parameter settings are also listed in Table 2.

2) RESULTS AND ANALYSIS

Simulation results with the generated trajectories are presented in Fig. 12. Similar to results in single USV path planning, obstacles are presented as irregular areas in black. The leader USV is displayed as a red dot whereas the two followers are coloured in blue and yellow, respectively. The destination area is also shown as the yellow area in the bottom right part in the environment. A successful mission will be

achieved when the leader USV arrives at the destination area. Fig. 12 (a)-(c) clearly show that collision free trajectories can be generated for the USV formation and the triangular shape can be well maintained throughout the trajectory. It should be noted that the generated trajectories may not be optimised with regard to the total distance, especially in Fig. 12 (a), where a curved path is generated before the formation reaches the destination area. This is mainly due to the strict requirement on retaining the formation shape as any short-cut may break the triangular shape.

TABLE 3. Actions taken by the USV formation for the simulation in Fig. 12(c).

Action	Leader USV	Follower USV 1	Follower USV 2
1	Turn right with 10°	Turn right with 10°	Turn right with 10°
27	Turn left with 10 degrees	Turn left with 10°	Turn left with 10°
27	Turn left with 10°	Turn left with 10°	Turn left with 10°
27	Turn left with 10°	Turn left with 10°	Turn left with 10°
27	Turn left with 10°	Turn left with 10°	Turn left with 10°
27	Turn left with 10°	Turn left with 10°	Turn left with 10°
27	Turn left with 10°	Turn left with 10°	Turn left with 10°
1	Turn right with 10°	Turn right with 10°	Turn right with 10°
1	Turn right with 10°	Turn right with 10°	Turn right with 10°
27	Turn left with 10°	Turn left with 10°	Turn left with 10°
1	Turn right with 10°	Turn right with 10°	Turn right with 10°

The formation maintainability can be further quantitatively proved by recording the actions taken by the USV formation at each time step. As shown in Table 3, the same actions can be also selected by each USV within the formation which lead to an unified and coordinated maneuver. However, it should also be noted that such a coordinated behaviour may be invalid. Take Fig. 12 (d) for example, in an environment with densely located obstacles, the algorithm fails to find safe trajectories, causing the leader USV to collide with an obstacle *en route*. To address this issue, it therefore becomes important to adopt a new algorithm which enables a flexible formation shape to accommodate complex environments.

Fig. 13 shows the moving average episode reward with a moving window size of 1000 episodes for the map shown in Fig. 12 (c). Due to the increased space and action dimensions, it requires much more training episodes for the agent to converge to a policy with reasonable performance. The training was terminated at episode 1×10^5. Nevertheless, for all the four maps investigated, similar reward trends have been observed.

C. USV FORMATION PATH PLANNING WITH FLEXIBLE SHAPE

This section presents the results of USV formation path planning with a flexible formation shape. Key criterion including the capability of finding the optimal trajectory reaching the destination, navigation safety along the route, formation shape variation (in constrained areas) and formation shape maintenance (in open-space areas) have been assessed. By undertaking simulations in the same environments as previous section, results are compared to show an improved navigation performance provided by using a flexible formation shape strategy. In addition, to further validate the algorithm's capability, a simulation in a practical maritime environment, Portsmouth harbour, UK, has been conducted.

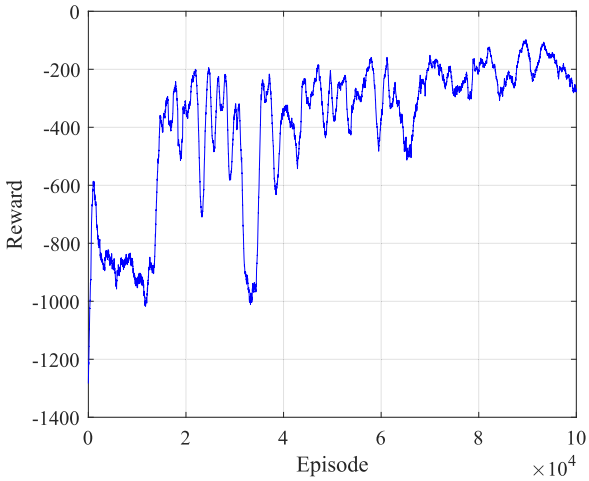


FIGURE 13. USV constant shape formation training process of Fig. 12 (c): moving average episode reward with a moving window size of 1000 episodes.

TABLE 4. Agent training parameters for USV flexible shape formation.

Parameter	Value
Speed	50 pixels per step
Learning rate	0.07
Discount rate	0.9
ε	0.1
Memory size	3×10^5
Mini-batch size	64
$\hat{Q}$ update frequency	3×10^3 steps
NN hidden layer 1	20 neurons
NN hidden layer 2	27 neurons

1) TRAINING SETTINGS

Table 4 presents the agent training parameter settings for the USV flexible shape formation path planning algorithm. Considering the state space dimension is still 9, i.e., the Q-network input feature number is the same as in Section IV-B, the Q-networks are designed the same as in Section IV-B. The USV speed is also 50 pixels per step to further the algorithm's capability of maintaining the formation shape but with flexibility of adjustments whenever necessary. Since more complex reward functions have been designed to realised flexible formation in this section, the agent would typically need 6×10^5 to 1×10^6 to find optimal trajectories. Also, the replay memory size is further increased to 300 000.

2) RESULTS AND ANALYSIS

Simulation results of USV formation path with a flexible formation shape is shown in Fig. 14. The used simulation environments are the same to those in Section IV-B to mainly reflect the improved performance obtained by using the flexible shape strategy. In Fig. 14 (a), it shows that a shorter trajectory can be generated compared to that in Fig. 12 (a). This is mainly because the USV formation is able to learn

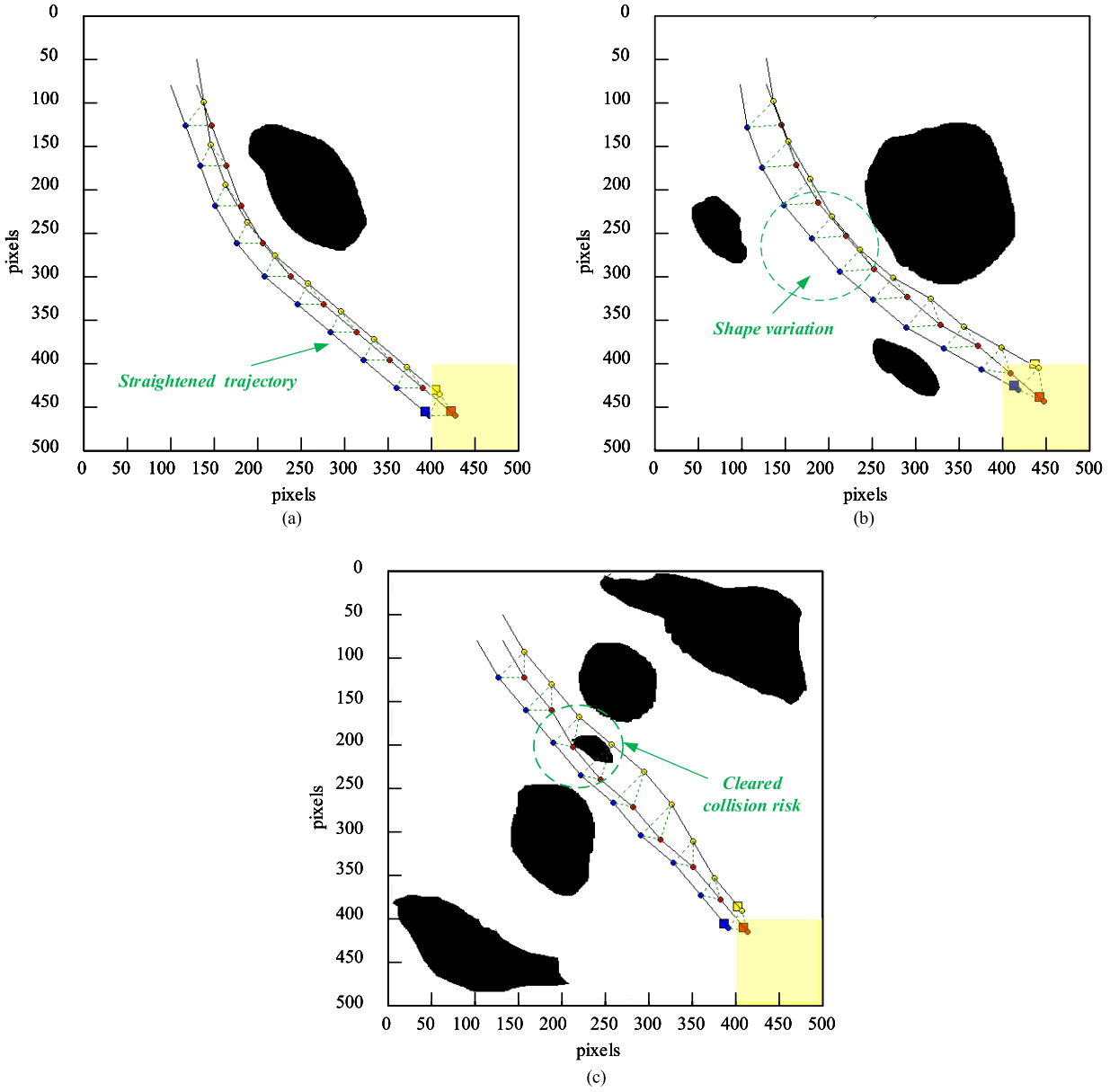


FIGURE 14. Simulation results for USV formation path planning algorithm with a flexible formation shape. Same simulation environments have been used but with different results: (a) a shorter trajectory can be generated compared to the result in Fig. 12 (a), (b) an improved safety margin can be obtained compared to the result in Fig. 12 (b) and (c) the collision can be well cleared compared to the result in Fig. 12(c).

a better navigation strategy, where the shape of the formation is varied in initial stages and a straight trajectory with the shortest distance to the destination area can be obtained.

Such a profound learning capability is further validated in more complex environments as shown in Fig. 14 (b) and (c). For example, in Fig. 14 (b), in order to ensure safety in a relatively constrained environment with multiple obstacles, the USV formation is able to learn that by varing the formation, a safer distance can be kept between the USVs and the obstacles (as shown as the circled area), which is in contrast to the result in Fig. 12 (b).

More importantly, the adoption of a flexible formation shape can resolve the dilemma that is encountered by using fixed formation shape strategy. As shown in Fig. 14 (c), where the fixed shape path planning strategy fails to find a route, by varying the formation shape (splitting the formation before approaching the obstacle), the potential collision occurred to the leader USV can be cleared. In addition, after collision has been cleared, the formation is able to merge together to retain the previous shape, which proves a high-level intelligence provided by the proposed algorithm.

Fig. 15 shows the reward trend over the training process of the map in Fig. 14 (c). In the first 1×10^5 training

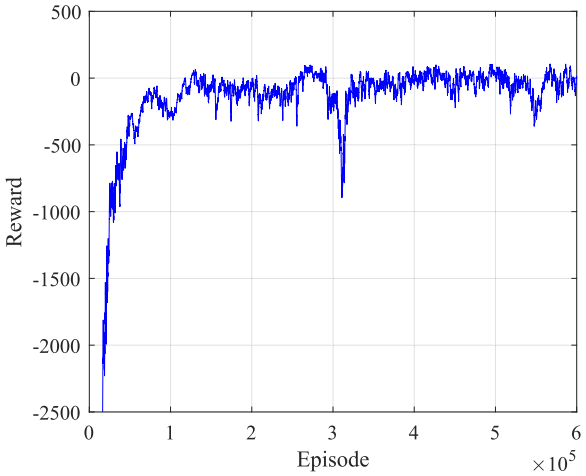


FIGURE 15. 3-USV flexible shape formation training process Fig. 14 (c): moving average episode reward with a moving window size of 1000 episodes.

episodes, the mean of episode reward increased rapidly to a value around −100. This due to the magnitude of collision avoidance reward is far greater than other rewards; the agent managed to avoid collisions in this period. From episode 1×10^5 to 4×10^5, the mean episode reward kept increasing in general by exploiting gained experience but also fluctuated due to explorations. From episode 4×10^5 onwards, the mean reward value converged to a value around 0, suggesting the training has converged. Note that similar convergence trends have been observed for the other three maps presented in Fig. 14.

The final set of test in this section has been carried out in a simulation environment extracted from an area in Portsmouth harbour, UK (shown in Fig. 16 (a)). The selected area contains both open sea area as well as narrow waterways, which is ideal to validate the proposed algorithm's capability of adaptively varying formation shape. The results are shown in Fig. 16 (b). It clearly shows that USVs starts the mission with a triangular shape and such a formation has been adaptively varied and shrinks into a smaller dimension to accommodate the constraints imposed by a narrow channel. Once the USV formation travels through the channel, the formation shape expands to its previous configuration.

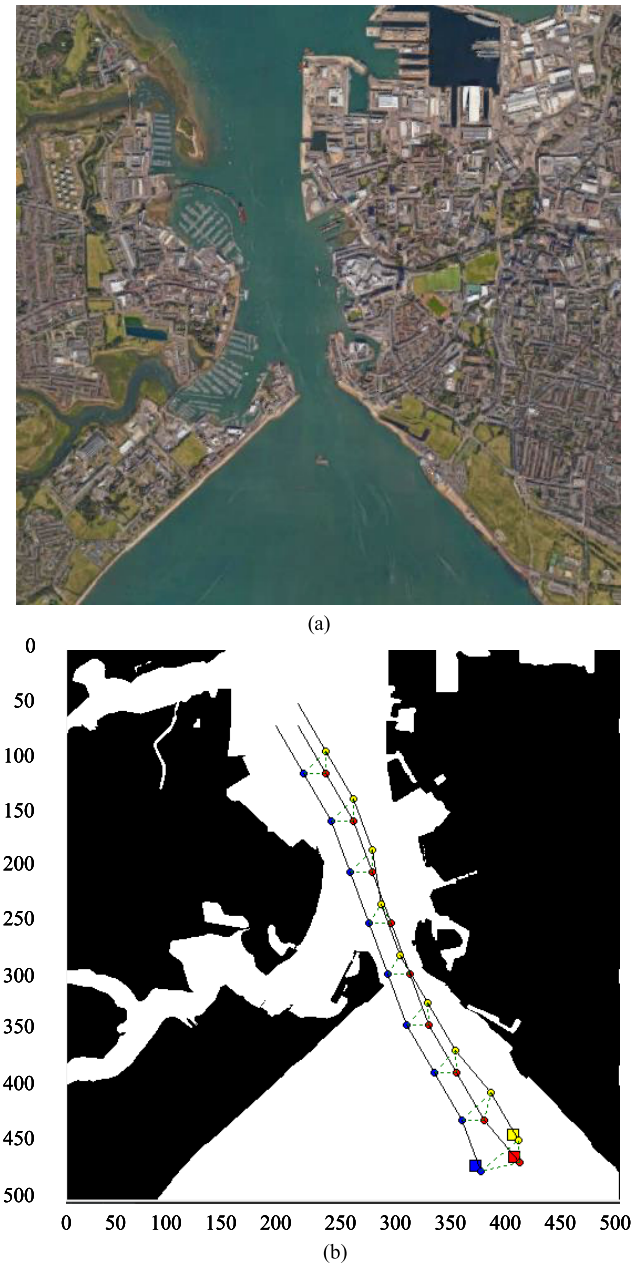


FIGURE 16. USV formation path planning in Portsmouth harbour area. (a) Simulation area in Portsmouth harbour. (b) Learned trajectory for USV formation using flexible shape strategy.

D. ALGORITHM COMPARISON IN A PRACTICAL MARITIME ENVIRONMENT

Finally, to compare two formation path planning algorithms, i.e., with formation maintenance capability versus with formation variation capability, a test is carried out in a selected area close to Plymouth harbour, UK. The results are presented in Fig. 17. The formation algorithm with shape maintenance capability aims to generate the collision free trajectory and maintain the formation shape; whereas the algorithm with flexible shapes aims to generate the shortest collision free trajectory to the destination area. Apart from a demanding requirement for obstacle environment of constant formation shape algorithm, the trajectory length generated by the latter algorithm may be longer than the former one under the same simulation environment. By comparing the results in Fig. 17, the trajectory of the leader USV generated by flexible formation shape algorithm (Fig. 17 (a)) is relatively straight towards the destination area without being restricted by formation condition. However, for the formation maintenance algorithm, the heading direction of the leader USV may keep changing so that the two followers can follow easily,

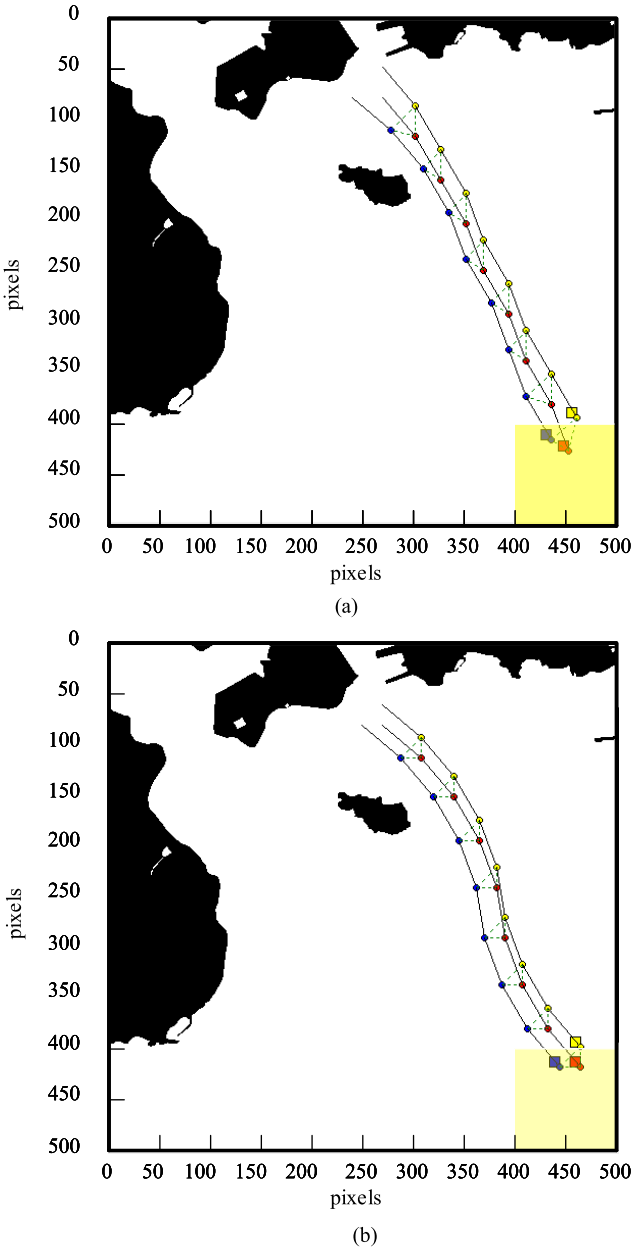


FIGURE 17. USV formation path planning in Plymouth harbour area. (a) Results of formation path planning with formation maintenance capability. (b) Results of formation path planning with formation variation capability.

indicating that the motion of leader USV will be restricted by formation shape as well as the navigating condition of the two follower vessels.

V. CONCLUSION AND FUTURE WORK

In this paper, a deep reinforcement learning (Deep Q network) based path planning algorithm has been developed for USVs and USV formations. Two new Markov decision processes (MDPs) are proposed to mathematically describe USV and USV formation path planning problems. Specifically, USV's kinematic motion is integrated into the USV path planning MDP together with an improved collision risk assessment strategy to provide better navigation results. Meanwhile, for USV formation path planning, two critical features (maintaining fixed formation shape and flexibly varying formation shape) have been addressed by designing two reward functions to encourage the learning of these two behaviours via proper training processes. The proposed algorithms have been validated in both self-constructed simulation environments and areas extracted from practical maritime environments.

In terms of future work, improved versions of MDPs for both single USV and USV formation path planning can be developed. At present, the simulation environments have been assumed to be a calm sea state without environment influences. Such an assumption may lead to compromised learning performances in a severe ocean environments, where the ever-changing current, winds and tides constantly impact the vessels. Real-time environment information can be incorporated into MDPs in future work and the robustness of the deep learning algorithm will be further investigated.

Another interesting direction for future work is to investigate the action space. Although DQN is capable of dealing with continuous state space and will become invalid for solving the problem with continuous action space. Currently, although the action space defined for USV path planning has already taken the vehicle's motion constraint into consideration with a refined set of actions, a more realistic approach would be to apply a continuous action space, which will facilitate the direct control of autopilot of vehicle. To achieve such a function, new deep reinforcement training processes such as Deep Deterministic Policy Gradient (DDPG), Asynchronous Advantage Actor-Critic Algorithm (A3C) or Soft Actor-Critic (SAC) can be employed in the future work.

REFERENCES

[1] Y. Liu and R. Bucknall, "Path planning algorithm for unmanned surface vehicle formations in a practical maritime environment," *Ocean Eng.*, vol. 97, pp. 126–144, Mar. 2015.

[2] Y. Liu and R. Bucknall, "The angle guidance path planning algorithms for unmanned surface vehicle formations by using the fast marching method," *Appl. Ocean Res.*, vol. 59, pp. 327–344, Sep. 2016.

[3] Y. Singh, S. Sharma, R. Sutton, D. Hatton, and A. Khan, "A constrained A* approach towards optimal path planning for an unmanned surface vehicle in a maritime environment containing dynamic obstacles and ocean currents," *Ocean Eng.*, vol. 169, pp. 187–201, Dec. 2018.

[4] Y. Liao, Z. Jia, W. Zhang, Q. Jia, and Y. Li, "Layered berthing method and experiment of unmanned surface vehicle based on multiple constraints analysis," *Appl. Ocean Res.*, vol. 86, pp. 47–60, May 2019.

[5] Y. Liu, R. Bucknall, and X. Zhang, "The fast marching method based intelligent navigation of an unmanned surface vehicle," *Ocean Eng.*, vol. 142, pp. 363–376, Sep. 2017.

[6] C. Chen, X.-Q. Chen, F. Ma, X.-J. Zeng, and J. Wang, "A knowledge-free path planning approach for smart ships based on reinforcement learning," *Ocean Eng.*, vol. 189, Oct. 2019, Art. no. 106299.

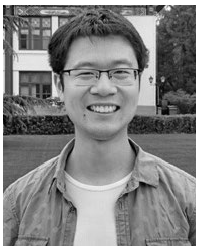
[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*. [Online]. Available: https://arxiv.org/abs/1312.5602

[8] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, and A. Bolton, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[9] J. Woo, C. Yu, and N. Kim, "Deep reinforcement learning-based controller for path following of an unmanned surface vehicle," *Ocean Eng.*, vol. 183, pp. 155–166, Jul. 2019.
[10] K. Jin, H. Wang, and H. Yi, "End-to-end trajectory tracking algorithm for unmanned surface vehicle using reinforcement learning," in *Proc. 29th Int. Ocean Polar Eng. Conf.*, 2019, pp. 1–7.
[11] J. Woo, J. Park, C. Yu, and N. Kim, "Dynamic model identification of unmanned surface vehicles using deep learning network," *Appl. Ocean Res.*, vol. 78, pp. 123–133, Sep. 2018.
[12] B. Yoo and J. Kim, "Path optimization for marine vehicles in ocean currents using reinforcement learning," *J. Mar. Sci. Technol.*, vol. 21, no. 2, pp. 334–343, Jun. 2016.
[13] Y. Cheng and W. Zhang, "Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels," *Neurocomputing*, vol. 272, pp. 63–73, Jan. 2018.
[14] R. S. Sutton and A. G. Barto, *Reinforcement Learning—An Introduction*. Cambridge, MA, USA: MIT Press, 1998.
[15] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, Cambridge, U.K., 1989.
[16] X. Zhang, C. Wang, Y. Liu, and X. Chen, "Decision-making for the autonomous navigation of maritime autonomous surface ships based on scene division and deep reinforcement learning," *Sensors*, vol. 19, no. 18, p. 4055, 2019.
[17] H. Xu, N. Wang, H. Zhao, and Z. Zheng, "Deep reinforcement learning-based path planning of underactuated surface vessels," *Cyber-Phys. Syst.*, vol. 5, no. 1, pp. 1–17, 2019.
[18] R. Song, Y. Liu, and R. Bucknall, "Smoothed A* algorithm for practical unmanned surface vehicle path planning," *Appl. Oceans Res.*, vol. 83, pp. 9–20, 2019.
[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.
[20] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," *Cited*, vol. 14, p. 8, Feb. 2012.

XINYUAN ZHOU received the B.Eng. degree in electrical and electronic engineering from the University of Liverpool and the M.Sc. degree in power system engineering from the University College London (UCL). Her research interests include artificial intelligence, reinforcement learning, path planning, and robotics.

PENG WU received the B.Eng. degree in marine engineering from Shanghai Maritime University, in 2011, and the M.Sc. degree in marine engineering from University College London (UCL), in 2015, where he is currently pursuing the Ph.D. degree. He worked on marine propulsion systems as a Field Engineer for nearly four years. His research focuses on hybrid fuel cell/battery propulsion system design and intelligent energy management strategies with route optimization.



HAIFENG ZHANG received the B.S. degree in computer science and the bachelor's degree in economics from Peking University, in 2012, and the Ph.D. degree from Peking University, in 2018, under the supervision of Prof. W. Li. He is currently a Research Fellow with the Department of Computer Science, University College London (UCL). Working with Prof. J. Wang, his main research areas include reinforcement learning, game AI, game theory, and computational advertising.

WEIHONG (GRACE) GUO (M'18) received the B.S. degree in industrial engineering from Tsinghua University, Beijing, China, in 2010, and the M.S. and Ph.D. degrees in industrial and operations engineering from the University of Michigan, Ann Arbor, MI, USA, in 2012 and 2015, respectively.

She is currently an Assistant Professor with the Department of Industrial and Systems Engineering, Rutgers University. Her research interests include manufacturing data analytics, process monitoring, anomaly detection, quality evaluation, and system informatics.

Dr. Guo is a member of the Institute of Industrial and Systems Engineers (IISE), the Institute for Operations Research and the Management Sciences (INFORMS), the American Society of Mechanical Engineers (ASME), the IEEE Robotics and Automation Society, and Tau Beta Pi.

YUANCHANG LIU received the B.Eng. degree in control engineering from the Dalian University of Technology, in 2010, and the M.Sc. degree in power systems engineering and the Ph.D. degree in marine control engineering from the University College London, in 2011 and 2016, respectively. He was a Research Fellow in robotic vision and autonomous vehicles with the Surrey Space Centre, University of Surrey. He is currently a Lecturer with the Department of Mechanical Engineering, University College London. His research interests include automation and autonomy, with a special interest in the exploration of technologies for sensing and perception, and guidance and control of intelligent and autonomous vehicles. He is currently supervising four Ph.D. students, working on the project of autonomous navigation for unmanned marine vehicles.