A centralised training algorithm with D3QN for scalable regular unmanned ground vehicle formation maintenance

Hao Yuan  |  Jun Ni  |  Jibin Hu

School of Mechanical Engineering, Beijing Institute of Technology, Beijing, People’s Republic of China

Correspondence
Jun Ni, School of Mechanical Engineering, Beijing Institute of Technology, Beijing, People’s Republic of China.
Email: nijun_bit@163.com

Abstract
The unmanned ground vehicle (UGV) has been widely used to accomplish various missions in civilian or military environments. Formation of the UGVs group is an important technique to support the broad applications of multi-functional UGVs. This study proposes a scalable regular UGV formation maintenance (SRUFM) algorithm based on deep reinforcement learning (DRL), which aims to use a unified DRL framework to improve the lateral control and longitudinal control performance of UGV in different situations of the formation. Based on the tailored design of state, action space, related inter-vehicle information, and the dueling double deep Q-network (D3QN), SRUFM outperforms nature deep Q-network and double deep Q-network in the exploration efficiency and the convergence speed in same CARLA training environments with fixed size formation. Furthermore, when the formation’s scale is extended with similar initialisation conditions, the SRUFM can still get a nearly 90% success rate to execute all experiment formation maintenance missions after 4000 episodes of training. Each UGV in the formation can keep distance within the upper and lower error threshold of 0.15 m. The simulation experiments show that the proposed centralised training frame with D3QN is suitable to solve scalable regular UGV formation maintenance missions.

1 | INTRODUCTION

Unmanned ground vehicle (UGV) is the vehicle with the ability of autonomous driving, which can replace humans to perform missions in different environments under autonomous or manual intervention modes [1]. Multi-UGVs collaboration is an essential technique to support the broad UGVs application [2]. The cooperation requires multi-UGVs to assist each other to achieve the best overall performance during the task. The challenge is to design an effective multi-agent framework for UGV formation control in various application environments. Nowadays, many multi-agent frameworks are proposed based on formation control algorithms, such as virtual structure method [3], behaviour-based method [4], leader-follower method [5], graph-based method [6], and potential field method [7]. However, these methods require experts’ knowledge to adjust the agents’ control parameters to meet different working scenarios’ requirements.

Recently, the deep reinforcement learning (DRL) method has been dramatically promoted and shows a great prospect to solve a variety of problems in the UGV area, including dynamic control [8], path planning [9], and motion harmonisation [10]. Compared to classic autonomous drive systems with environment perception, path planning, and dynamics control modules [11], the DRL method combines deep neural networks with a reinforcement learning frame. This combination can help UGV achieve a faster response to more complex traffic scenarios by learning strategies from high-dimensional perceptual input through the end-to-end model [12]. Therefore, the application of DRL in the UGV field is attracting an increasing research focus.

The formation control problems for multi-UGVs need to consider the performance of both lateral and longitudinal control. Specifically, the formation maintenance mission can generally be divided into lane-keeping mission in lateral control and car-following mission in longitudinal control.
The lane-keeping mission is supposed to improve the path tracking performance and handling stability in the lane-keeping manoeuvre. Besides, the control methods can be classified as vehicle model-based methods [13–17] and vehicle model-free methods [18–20]. In the vehicle model-based methods, Kim et al. proposed a torque overlay-based robust steering control technique that consisted of a non-linear damping controller [21]. Kang et al. proposed a novel multi-rate lane-keeping system by using a kinematic-based model, while the look-ahead output measurement matrix and the multi-rate Kalman filter were applied to increase the damping of the vehicle and resolve the asynchronous sampling time of multi-sensors [22]. In the vehicle model-free methods, Fehrer et al. presented the application of a Q-learning reinforcement learning in the lane-keeping mission using track curvature, lateral position, and relative yaw angle as inputs. The latter distance error could be controlled within −0.4 to 0.6 m after training [23]. Also, Aradi et al. presented the research using a policy gradient-based reinforcement learning method for vehicle lane-keeping in a simulated highway environment [24]. On the other hand, the car-following models are mainly designed to control the vehicle’s acceleration in the platoon control. The traditional car-following models can be classified as stimulus-response models [25], safe distance models [26], psychophysical models [27], and desired-measure models [28]. Due to high-fidelity traffic data availability, data-driven methods have been widely applied to solve car-following problems, as they combine additional parameters that influence agents’ behaviour to train more complex models [29]. Hongfei et al. proposed a four-layer neural network to predict the follower’s acceleration [30]. Khodayari et al. proposed a modified neural network approach with one hidden layer to simulate and predict the car-following behaviour based on the instantaneous reaction delay, reaction delay, relative speed, relative distance, and follower’s speed as the inputs [31].

Although the previous works have made many contributions, several points need to be improved as follows: (1) Most of the previous research only focused on solving lateral control or longitudinal control problems individually. Therefore, to guarantee UGV dynamics control’s performance, they need additional longitudinal control strategies to solve the lane-keeping missions or different lateral control strategies for solving the car-following missions. Obviously, the separate control system vastly increases the complexity of the entire control system. (2) Most of the previous works are not efficient for solving the formation maintenance problem. For vehicle model-based methods, the more the UGVs are considered, the larger the dynamics parameters are needed to be adjusted when the types of UGVs are changed. As for vehicle model-free methods, some multi-UGVs methods with reinforcement learning (RL) are proposed, such as value decomposition networks [32] and monotonic mixing network [33]. However, they need to train an individual network for each UGV, which significantly increases the training time.

To this end, this study attempts to design an efficient UGV formation maintenance training algorithm by considering the lateral and longitudinal controls at the same time. Scalable regular UGV formation, which can be divided into several similar UGV formation units according to the form characteristic of the regular formation, will be mainly studied in this work. This study’s main contributions are as follows: (1) Considering the separate control system’s complexity, the dueling double deep Q-network (DDQN) structure-based algorithm under comprehensive consideration of both lateral and longitudinal controls is proposed. Deep networks combine the commonly used inputs from model-free lane-keeping methods and data-driven car-following methods. Compared with other value-based RL methods (nature deep Q-network (Nature DQN) and double DQN (DDQN)), the algorithm designed to solve scalable regular UGV formation maintenance (SRUFM) mission outperforms in the efficiency of exploration and the speed of convergence. It also keeps a high success rate to finish the experiment route. (2) SRUFM aims to offer an efficient idea to solve large multi-UGVs formation maintenance missions with less knowledge, computer power, and time cost. In the training part, only one of the smallest formation units is needed. Each UGV in the unit is learned by a centralised SRUFM. After the training, more UGVs with similar position information can be expanded behind the unit to assemble a larger formation. Utilising the same action space, episode ending rules, and reward mechanism, SRUFM still has a high success rate (around 90%) to finish the larger formation maintenance in most formation types without additional training.

The remainder of this article is organised as follows. In Section 2, the experimental environment settings in the CARLA simulator are described. In Section 3, the SRUFM learning method for UGV formation maintenance problems is proposed. In Section 4, different training and testing formation maintenance simulation cases under SRUFM are carried out to compare with other DRL methods. In Section 5, conclusions and future works are presented.

## 2 | EXPERIMENTAL ENVIRONMENT

The classic autonomous system of UGV can be divided into three parts: Environment perception and localisation, path planning, and dynamics control. The environment perception part analyses information about the surrounding environment and UGV itself by processing data from lidar, camera, and other sensor components. The path planning part selects the optimal decision and makes the optimal path under careful consideration of perception results. The dynamics control part tracks the theoretical optimal path and calculates the actual control signals of steering, drive force, and brake force to realise the movement of the UGV. However, the end-to-end autonomous system integrates environment perception, path planning, and dynamics control into one learning framework.

In this study, CARLA simulator [34] is applied to train and test the performance of SRUFM method, which can simulate hardware configuration and autonomous system of UGV. A section of straight road without other dynamic element is chosen as the training route. Red-green-blue (RGB) camera, collision sensor, and lane invasion sensor are selected to provide perspective images, collision detection, and lane line type
recognition at each frame time for each UGV. More detailed simulation environment settings about regular formation types, action space, episode ending rules, and reward mechanisms are explained below:

### 2.1 Regular Formation Type

SRUFM is supposed to solve the regular formation maintenance mission, where the formation can be divided into several smallest training units. There are three regular formations, and their regular smallest training units (inside the dashed box) are shown in Figure 1. In the training part, one of the smallest units is selected as the training object. One UGV as the leader (L) and the other UGV as the follower (F). In the testing part, more UGVs can be added as followers to achieve the larger-scale formation.

### 2.2 Action space for the aim

According to the formation maintenance mission requirement, four actions are designed to form the action space shown in Table 1. In CARLA, by normalising the throttle value to [0, 1], the brake value to [0, 1], and the steering to [–1, 1], three unified control variables for different types of UGVs are designed. The specific aim of experiments is to teach the leader UGV to keep the expected distance from the leader UGV after training.

### 2.3 Episode ending rules

Four rules are designed to judge whether the current episode should be over as shown in Table 2. Once one of the UGVs has reached one collision or driven out of the lane, and if both UGVs reach the end, the end of the episode will be triggered. Additionally, to prevent the simulation from falling into a dead zone, the episode will also end once the duration exceeds the maximum time. Each time the current episode is over, CARLA simulator will initialise UGVs’ position and start a new episode.

### 2.4 Reward mechanism

The setting of the reward mechanism plays a vital role in the performance of SRUFM method. After each step, the UGVs’ rewards will be calculated according to the status of each UGV. The reward mechanism is slightly different between the leader and the follower due to the different additional missions of each UGV. The rewards about aim (raim), expected velocity (rcv), current velocity (rv), and leader reachability (rll) are designed for the leader. The rewards about expected distance (rred) and follower reachability (rf) are designed for the follower. However, the reward for an accident (racc) is shared for both UGVs. When the UGV collides with any element or crossing lane line, racc is set to –100. In other cases, racc is set to 0.5.

The additional mission of the leader UGV is to reach the expected velocity as soon as possible (in this study, the expected velocity is consistent with the maximum speed that the action space can achieve). The main role of raim is to urge the leader UGV to move as far as possible from the starting point. At each time step t, the longitudinal distance d(t) between the current location of the UGV and the starting point will be calculated. raim is set according to the value of d(t) – d(t – 1). If this value is larger than 0, raim = 2; otherwise, raim = –1.5. rv is set as v(t)/vt, where v(t) is the current velocity and vt is the expected velocity. The expected velocity range is set as [vt – 1, vt + 1], and when the speed of the leader is within this range, rrv is set as 1. In other cases, rrv is set as –1.5. If leader UGV or follower UGV reaches the destination, rrd will be set as 1000. In other cases, rrd is set as 0. At each step, the total reward of the leader UGV is r = rrv + raim + rcv + rrd + rll. The additional mission of the follower UGV is to keep the expected distance. At each time step t, the relative distance between the leader UGV and the follower UGV D(t) will be calculated. The expected D(t) range is set as [Dt – 0.15, Dt + 0.15]. When D(t) is within this range, rrd is set as 1. In other cases, rrd is set as –1.5. If follower UGV reaches the destination, rrd will be set as

---

**TABLE 1** Action space

| Action of leader unmanned ground vehicle (UGV) | Action of follower UGV | [Throttle, steer, brake] |
|-----------------------------------------------|------------------------|-------------------------|
| ac(L0)                                        | ac(F0)                 | [0.6, 0.0, 0.0]         |
| ac(L1)                                        | ac(F1)                 | [0.2, 0.0, 0.0]         |
| ac(F2)                                        | ac(F2)                 | [0.4, -0.4, 0.0]        |
| ac(L3)                                        | ac(F3)                 | [0.4, 0.4, 0.0]         |

**TABLE 2** Episode ending rules

| Related sensor | UGV | End information       |
|----------------|-----|-----------------------|
| Collision sensor | L   | Collision_flag_l = true |
| L              | F   | Collision_flag_f = true |
| Lane invasion sensor | L   | Lane_flag_l = true     |
| L              | F   | Lane_flag_f = true     |
| Reach signal   | L   | Reach_flag_l = true    |
| L              | F   | Reach_flag_f = true    |
| Time signal    | L   | \( \Delta T \) > 25 s |
| F              |     |                       |
In other cases, \( r_{fr} \) is set as 0. At each step, the total reward of the follower UGV is \( r_{fi} = r_{ac} + r_{co} + r_{fr} \).

3 | SRUFM FOR UGV FORMATION MAINTENANCE

To suppress the overestimation problem and enhance the dynamic environmental information sensitivity, SRUFM retains the structure of D3QN. The online network and the target network still exist, and both networks adopt the dueling structure as shown in Figure 2. SRUFM solves scalable regular UGV formation maintenance mission by training one of the smallest units. Different from other RL-based multi-UGVs methods, the state of each UGV in the unit will be described separately. Each UGV has its state information, including the front camera image, relative speed, and relative distance information. Any of the UGVs in the unit uses the unified SRUFM for its action output in every step. SRUFM utilizes all UGVs’ experience to update and learn online as shown in Figure 3.

3.1 | The online and target neural network structure

In this study, the dueling network structure is used both in the online and target neural network as shown in Figure 2. The size of the image in the form of (length, width, depth) is defined. When acquiring the current environment information of UGVs from each frame, RGB-alpha format image obtained from the
CARLA camera should be changed into RGB format at first. Then the size of the new format image will be adjusted to \((320, 124, 1)\) according to the region of interest. Considering the possible influence of the changes between the two frames on the UGV decision-making, the grayscale images of two consecutive frames are stacked. The resulting image will be greyed to be one of the inputs, whose size is \((320, 124, 2)\). Detail layers parameters in this part are described below.

The first hidden layer is the convolution layer containing 32 filters of kernel size \((8, 8)\) with a stride of \(0\), which can reduce the features’ dimensions and extract the main features. The third hidden layer is the convolutional layers containing 64 filters of kernel size \((5, 5)\) while the fifth hidden layer containing 64 filters of kernel size \((4, 4)\), whose strides are both \(0\) and activation functions are the ReLu. The fourth and sixth hidden layers are max-pooling layers of size \((3, 3)\), whose strides are both \(2\). Relative speed and relative distance information as \((\Delta v, \Delta D)\) are combined with flattening array (for the leader UGV, \(\Delta V = v_t - v_j, \Delta D_t = 0\); for the follower UGV, \(\Delta V = v_t - v_j, \Delta D_t = D(t)\)). The network is then divided into a value function flow and an advantage function flow. The first layer in the two streams is a fully connected layer with 64 units, and the activation function is the ReLu. The second layer in the value function flow and the advantage function flow are added to get the value of the actions.

### 3.2 Algorithm design

The formation maintenance mission of UGVs can be defined as a Markov decision process (MDP). Each MDP problem can be described by a tuple \((S, A, P, R, \gamma)\), where \(S\) represents all possible states, \(A\) represents all possible actions, \(P\) represents the states transition function, \(R\) represents immediate rewards, and \(\gamma\) represents the discount factor. At each time step \(t\), while in state \(s_t\), the agent takes an action \(a_t\) by using policy \(\pi\). The next state \(s_{t+1}\) is determined accordingly by the transition function \(P\), and an immediate reward \(r_t\), using cumulative discounted rewards \(G_t\), to assess the current state value \(s_t\):

\[
G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}
\]

where \(t\) is the time step, \(k\) is the number of each step from the beginning of the current step \(t\) to the end of the episode, \(\gamma \in (0, 1)\) is the discount factor, and \(R_{t+k+1}\) is the immediate reward. Two value functions are formulated to represent the worth of the control action selection. They are named as a state-value function \(v_\pi(s_t)\) and state-action function \(q_\pi(s_t, a_t)\):

\[
v_\pi(s_t) = E_\pi \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | s_t \right]
\]

\[
q_\pi(s_t, a_t) = E_\pi \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | s_t, a_t \right]
\]

where \(\pi\) is a special control policy, \(s_t\) is the current state and \(a_t\) is the current action. Substituting Equations 1 with 3 to obtain the Bellman equation, we have

\[
q_\pi(s_t, a_t) = E_\pi \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | s_t, a_t \right]
\]

\[
= E_\pi \left[ R_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1}) | s_{t+1}, a_{t+1} \right]
\]

where \(s_{t+1}\) is the next state and \(a_{t+1}\) is the next action. From Equation 4, it can be seen that the value of \(q_\pi(s_t, a_t)\) is determined by immediate reward \(R_{t+1}\) and the Q-value of the next step \(q_\pi(s_{t+1}, a_{t+1})\). MDP aims to find the strategy that maximises \(q^*\).

\[
q^*(s_t, a_t) = \max_\pi q_\pi(s_t, a_t)
\]

In formation maintenance mission, the state transition function \(P\) that is used to calculate the accurate \(q_\pi(s_t, a_t)\) or \(V_\pi(s_t)\) is recursive to be figured out. Therefore, \(P\) approximated methods are needed. Based on temporal-difference (TD) methods, Q-learning [37] is widely used:

\[
q_\pi(s_t, a_t) = q_\pi(s_t, a_t) + \alpha [R_{t+1} + \gamma \max_a q_{\pi+1}(s_{t+1}, a_{t+1}) - q_\pi(s_t, a_t)]
\]

Q-learning uses a greedy policy to get the next action \(a_{t+1}\) of next state \(s_{t+1}\), where \(q_\pi(s_t, a_t)\) is online Q-value and \(q_{\pi+1}(s_{t+1}, a_{t+1})\) is target Q-value; \(\alpha\) represents a learning rate. However, since the formation maintenance environment’s state space is uncountable, it is necessary to construct a mapping between the corresponding state and the Q-value for each action.

\[
Q(s, a) = f(s, w)
\]

where \(f\) is the approximate function and \(w\) is the parameter of the function. Deep learning with neural networks is good at refining non-linear function relationships in complex application scenarios. Therefore, combining deep learning and Q-learning, the DQN method [38] was proposed. It uses the mean-square error to define the loss function \(L(\theta)\).

\[
L(\theta) = E[(R_{t+1} + \gamma \max_{a_t} q_{\pi}(s_{t+1}, a_{t+1}) | s_{t+1}, a_{t+1}) - q_\pi(s_t, a_t) | (s_t, a_t)]^2
\]

where \(\theta\) is the parameter of the deep neural network. The update of \(\theta\) is defined as

\[
\theta_{t+1} = \theta_t + \alpha [R_{t+1} + \gamma \max_{a_t} q_{\pi}(s_{t+1}, a_{t+1}) | s_{t+1}, a_{t+1}) - q_\pi(s_t, a_t) | (s_t, a_t)] \nabla_\theta q_\pi(s_t, a_t, \theta)
\]
The same neural network \( q_{\pi}(s,a;\theta) \) is used to calculate the online Q-value and the target Q-value, which will cause a bad convergence. An optimised DQN named Nature DQN [39] was proposed. It owns an online network to calculate online Q-value \( q_{\pi}(s_t,a_t;\theta) \) and the target network to calculate target Q-value \( q(s_{t+1},a_{t+1};\theta') \). The \( \theta_{t+1} \) is defined as

\[
\theta_{t+1} = \theta_t + \alpha [R_{t+1} + \gamma \max_{a_{t+1}} q(s_{t+1},a_{t+1};\theta')] - q_{\pi}(s_t,a_t;\theta) \tag{10}
\]

According to Equation 10, the maximum of the target Q-value is always used to be selected, which leads to the over-optimisation of the target value estimate. DDQN [40] network is designed to solve the target value over-optimisation problem by choosing the action that owns max Q-value from the online Q-network and its Q-value from the target Q-network as the target Q-value. In SRUFM, this idea is inherited. The loss function is defined as

\[
L(\theta) = E[(R_{t+1} + \gamma q(s_{t+1}, \arg \max_{a_{t+1}} q_{\pi}(s_{t+1}, a_{t+1};\theta') ; \theta')) - q_{\pi}(s_t,a_t;\theta)]^2 \tag{11}
\]

In SRUFM algorithm, dueling DQN [41] is the main structure of both two Q-networks, which separates the final Q-value into two parts: The state-action value \( V(s_t;\theta,\beta) \) and action-advantage value \( A(s_t,a_t;\theta,\alpha) \).

\[
q_{\pi}(s_t,a_t;\theta,\alpha,\beta) = V(s_t;\theta,\beta) + A(s_t,a_t;\theta,\alpha) \tag{12}
\]

where \( \theta \) denotes the parameter of convolutional layers, \( \alpha \) and \( \beta \) are two stream parameters of fully connected layers in a dueling network. However, the various combinations of values \( V \) and \( A \) can get the same Q-value, which reduces the algorithm’s stability. Therefore, in SRUFM, the Q-value of each UGV is designed as

\[
q_{\pi}(s_t,a_t;\theta,\alpha,\beta) = V(s_t;\theta,\beta) + A(s_t,a_t;\theta,\alpha) - \bar{A}(s_t,a_t;\theta,\alpha) \tag{13}
\]

where \( \bar{A}(s_t,a_t;\theta,\alpha) \) is the average value of the sum of output values of the advantage function. According to Equations 11 and 13, the loss function used in SRUFM is designed as

\[
L(\theta,\alpha,\beta) = E[(R_{t+1} + \gamma q(s_{t+1}, \arg \max_{a_{t+1}} q_{\pi}(s_{t+1}, a_{t+1};\theta,a',\beta') ; \theta') - q_{\pi}(s_t,a_t;\theta,\alpha,\beta)]^2 \tag{14}
\]

where \( s_t \) is described as \([m_t, g_t] \), \( m_t \) is the information of front camera image, and \( g_t \) is a two-dimensional vector composed of the relative speed and distance of each UGV.

The pseudo-code of SRUFM process is shown in Algorithm 1.

---

Algorithm 1: SRUFM

**Input:** Observation information agent \( i \) = \([s_i, a_i, g_i]\), \( s_i = [m_x, g_x] \).

**Output:** Weights \( \theta^*, \alpha^*, \beta^* \) for SRUFM networks.

Initialise the agent \( i \) = 1,2, initialise the experience replay repository \( D \) to capacity \( N \), the historical observations \( U \) = 0, the mini training replay memory size \( U_{\text{min}} \) = 1000, the max replay memory size \( U_{\text{max}} \) = 6000, the exploration rate \( \varepsilon \) = 1, Q-network and its parameters \( \hat{\theta}, \hat{Q} \)-network with \( \beta' = \beta \). The Q-value of each action could be obtained by \( Q = 1 + \alpha \).

for episode = 1 to \( M \) do

for \( t = 1 \) to \( T \) do

Get observation information \( q_{\pi}(s_t,a_t;\theta,\alpha,\beta) \).

Select a random \( a_{t+1}, \alpha_{t+1} \) with exploration rate \( \varepsilon \), otherwise select:

\[
\begin{align*}
\alpha_{t+1} &= \arg \max_{\alpha_{t+1}} q_{\pi}(s_{t+1}, a_{t+1};\theta,\alpha,\beta) \\
\beta_{t+1} &= \arg \max_{\beta_{t+1}} q_{\pi}(s_{t+1}, a_{t+1};\alpha,\beta,\beta')
\end{align*}
\]

Get \( (s_{t+1}, R_{t+1}, \text{done}) \) of each UGV from the environment and append in \( D \) with the tuple:

\[
(s_t, a_t, s_{t+1}, a_{t+1}, \text{done})
\]

if done

Break

\( \varepsilon = \varepsilon \times 0.995 \)

if \( U > U_{\text{max}} \)

Sample minibatch \( (s_t, a_t, s_{t+1}, R_t, \text{done}) \) from \( D \) and Update \( \theta, \alpha, \beta \) as:

\[
\begin{align*}
(\theta, \alpha, \beta) &= (\theta, \alpha, \beta) + \alpha [R_{t+1} + \gamma \max_{a_{t+1}} q_{\pi}(s_{t+1}, a_{t+1};\theta',\alpha',\beta'); \theta') - q_{\pi}(s_t,a_t;\theta,\alpha,\beta)] \\
\theta, \alpha, \beta &= \theta, \alpha, \beta
\end{align*}
\]

Reset the Q-network \( \beta', \alpha', \beta' = \theta, \alpha, \beta \) every C episode.

End for

Return \( \theta^*, \alpha^*, \beta^* \) for SRUFM network

---

A concise illustration of the frame structure and data flow of the SRUFM algorithm is shown in Figure 3. SRUFM is divided into two parts: Control and learning loops. In the control loop, at each step \( t \) each UGV inputs its respective images and relative information into the online Q-network and chooses the current action under the greedy policy. During the training, all experience is stored in the experience replay memory. In the learning loop, a certain amount of experience is extracted to generate the TD error for the online Q-network updating. The target Q-network synchronises parameters with the online Q-network every few episodes.

### 4 EXPERIMENTS

Choosing a suitable exploration-exploitation ratio makes it possible to get higher rewards efficiently. In SRUFM, the \( \varepsilon \)-greedy strategy [42] is used as the action is chosen to randomly select an action with the probability of \( \varepsilon \) and use the greedy
FIGURE 4  Action Q-values in three value-based RL methods of leader UGV and follower UGV in vertical formation mission

FIGURE 5  Action Q-values in three value-based RL methods of leader UGV and follower UGV in horizontal formation mission

FIGURE 6  Action Q-values in three value-based RL methods of leader UGV and follower UGV in diagonal formation mission
policy to get the action \( a = \text{argmax}_a(s, a; \theta) \) with the probability of 1–\( \varepsilon \). The probability of \( \varepsilon \) decreases from 1 to 0.001 at the speed of 0.995 at each step and finally remained at 0.001. The discount factor \( \gamma \) is set as 0.95, and the maximum time per episode is 25 s. With the hardware setup consisting of a single GTX 1050Ti GPU and 12 G memory, the max replay memory size \( U_{\text{max}} \) is set as 6000. During the experiments, the \( U_{\text{max}} \) is effective in preventing catastrophic forgetting and maintain training efficiency. Compared to the min training replay memory size \( U_{\text{min}} \) used by Wang et al. [43], the \( U_{\text{min}} \) is set as 1000 (the proportion of the \( U_{\text{min}} \) to the \( U_{\text{max}} \) is higher), which ensures the richness of samples in the early stage of training. SRUFM generates a replay memory sample at each step when the number of experience samples exceeds 1000. SRUFM network will be trained using 20 replay memory samples each time. The oldest sample will be replaced if the number of samples is over 6000. The target Q-network synchronises parameters with the online Q-network every 25 episodes. To compare SRUFM with Nature DQN and DDQN in an effective way, the action space is chosen as shown in Table 1. The total number of episodes is 4000. The training scenes are shown in Figures 7(a)–(c).

Different RL methods are used to learn three formations maintenance. The action Q-value changes are shown in Figures 4–6. It can be seen that the Q-value of \( \text{ac}(L0) \) and \( \text{ac}(F0) \) (where the throttle is equal to 0.6, steer and brake are equal to 0) are higher than other actions Q-values after all RL algorithms training, which meets the human driver’s decision-making in the same action space. It also indicates that centralised value-based reinforcement learning is effective in solving regular formation maintenance problems.
Compared to DDQN and Nature DQN, SRUFM performed higher efficiency in training the effective action \((L_0, F_0)\) as shown in Figures 4(a) and (e), 5(a) and (e), 6(a) and (e). This is especially true in the horizontal and diagonal formations mission, where the curve is more efficient in getting a stable distribution. The UGV rewards and distance varieties are shown in Figures 8 and 9. It can be seen that SRUFM outperforms other algorithms in the exploration efficiency and convergence speed. DDQN gets the close reward results around 2000 episodes in vertical formation (Figures 8(a) and (b)) and around 3000 episodes in horizontal formation (Figures 8(c) and (d)). In diagonal formation, although DDQN learns the stable reward around 3500 episodes (Figures 8(a) and (b)), the distance varieties are severe (Figure 9(c)) due to alternative not-so-good policy learned. DQN gets the close reward results around 3000 episodes in vertical formation (Figures 8(a) and (b)) and in horizontal formation (Figures 8(c) and (d)). In diagonal formation, it fails to learn high reward policy (Figures 8(e) and (f)). According to the gentle distance varieties, most episodes finish in a short time (Figure 9(e)).

To verify the algorithm’s generalisation ability, a similar CARLA environment is chosen to test different RL methods’ performance, as shown in Figures 7(d)–(f). Another follower UGV(UGVF1) is added to the formation. The maximum bias (Max) and minimum bias (Min) to the expected distance is calculated. The distance between leader UGV and follower UGV is \(D_1\), and the distance between the follower UGV and the new follower UGV is \(D_2\).

Without crossing lane situations or hits, each UGV in the formation can keep distance within the upper and lower error threshold of 0.15 m, which defines one successful episode. The number of total tests is 100, and the success rate is designed as \(n_s / 100\), where \(n_s\) is the total number of success.

The results shown in Table 3 are collected from a similar testing road. It can be seen that value-based RL methods are suitable to solve the regular formation maintenance problem. Besides, SRUFM has achieved a small stable bias and high success rate, while DDQN got a lower success rate in the vertical formation, and DQN fails to reach the destination in a diagonal formation (Figure 10).

**5 CONCLUSION**

This study presents a new algorithm, SRUFM, which uses DRL to solve scalable regular UGV formation maintenance prob-
ACKNOWLEDGEMENTS

The authors acknowledge the National Natural Science Foundation of China (51905036) and Beijing Natural Science Foundation (3202025) for financial support.

REFERENCES

1. Skrjicki, V., Šabanović, E., Žuraulis, V.: Autonomous road vehicles: Recent issues and expectations. IET Intel. Transport Syst. 14(6), 471–479 (2020)
2. Hu, G., et al.: Cooperative bypassing algorithm for connected and autonomous vehicles in mixed traffic. IET Intel. Transport Syst. 14(8), 915–923 (2020)
3. Abbasi, Y., Moosavian, S.A.A., Novinzadeh, A.B.: Formation control of aerial robots using virtual structure and new fuzzy-based self-tuning synchronization. Trans. Inst. Meas. Control. 39(12), 1906–1919 (2017)
4. Lee, G., Chwa, D.: Decentralized behavior-based formation control of multiple robots considering obstacle avoidance. Intell. Serv. Rob. 11(1), 127–138 (2018)
5. He, S., et al.: Leader–follower formation control of USVs with prescribed performance and collision avoidance. IEEE Trans. Ind. Inf. 15(1), 572–581 (2018)
6. Chu, X., et al.: Decentralised consensus-based formation tracking of multiple differential drive robots. Int. J. Control 90, 2461–2470 (2017)
7. Merheb, A.R., Gazi, V., Sezer-Uzol, N.: Implementation studies of robot swarm navigation using potential functions and panel methods. IEEE/ASME Trans. Mechatron. 21(5), 2556–2567 (2016)
8. Shan, Y., et al.: A reinforcement learning-based adaptive path tracking approach for autonomous driving. IEEE Trans. Veh. Technol. 69(10), 10581–10595 (2020)
9. Luviano, C.D., Yu, W.: Path planning of multi-agent systems in unknown environment with neural kernel smoothing and reinforcement learning. Neurocomputing 233, 34–42 (2017)
10. Ko, B., et al.: Speed harmonisation and merge control using connected automated vehicles on a highway lane closure: A reinforcement learning approach. IET Intel. Transport Syst. 14(5), 947–957 (2020)
11. Martinez-Diaz, M., Soriguera, F., Perez, I.: Autonomous driving: A bird’s eye view. IET Intel. Transport Syst. 13(4), 563–579 (2019)
12. Duan, J., et al.: Hierarchical reinforcement learning for self-driving decision-making without reliance on labelled driving data. IET Intel. Transport Syst. 14(5), 297–305 (2020)
13. Kamat, S.: Lane keeping of vehicle using model predictive control. In: 2019 IEEE 5th International Conference for Convergence in Technology (I2CT), Mumbai, India, pp. 1–6 (2019)
14. Liu, W., Li, Z.: Comprehensive predictive control method for automated vehicles in dynamic traffic circumstances. IET Intel. Transport Syst. 12(10), 1455–1463 (2018)
15. Junwei, L., et al.: Backstepping sliding mode lane keeping control of lateral position error with dynamic of tire steering device. Optik 127(5), 2439–2443 (2016)
16. Hu, C., et al.: RISE-based integrated motion control of autonomous ground vehicles with asymptotic prescribed performance. IEEE Trans. Syst. Man Cybern.: Syst. 99, 1–13 (2019)
17. Hu, C., Chen, Y., Wang, J.: Fuzzy observer-based transitional path-tracking control for autonomous vehicles. IEEE Trans. Intell. Transp. Syst. 99, 1–11 (2020)
18. Hongbo, W., Li, C., Weihua, Z.: Lane-keeping control based on an improved artificial potential method and coordination of steering/braking systems. IET Intel. Transport Syst. 13(12), 1832–1842 (2019)
19. Bojarski, M., et al.: End to end learning for self-driving cars. arXiv:1604.07316 (2016)
20. Jantz, M., et al.: End-to-end race driving with deep reinforcement learning. arXiv:1807.02371 (2018)
21. Kim, W., Son, Y.S., Chung, C.C.: Torque-overlap-based robust steering wheel angle control of electrical power steering for a lane-keeping system of automated vehicles. IEEE Trans. Veh. Technol. 65(6), 4379–4392 (2016)
22. Kang, C.M., Lee, S., Chung, C.C.: Multirate lane-keeping system with kinematic vehicle model. IEEE Trans. Veh. Technol. 67(10), 9211–9222 (2018)
23. Feher, A., Aradi, S., Becsi, T.: Q-learning based reinforcement learning approach for lane keeping. In: 2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI), Budapest, Hungary, pp. 31–36 (2018)
24. Aradi, S., Becsi, T., Gaspar, P.: Policy gradient based reinforcement learning approach for autonomous highway driving. In: 2018 IEEE Conference on Control Technology and Applications (CCTA), pp. 670–675 (2018)
25. Sajdi, S., Mohamed, K.: Incorporating vehicle mix in stimulus-response car-following models. J. Traffic Transp. Eng. 3(3), 226–235 (2016)
26. Wang, H., et al.: Study on safety distance model of fleet based on vehicle communication. Procedia Soc Behav Sci. 96, 2698–2705 (2013)
27. Aghabayk, K., Sarvi, M., Young, W.: Including heavy vehicles in a car-following model: Modelling, calibrating and validating. J. Adv. Transp. 50(7), 1432–1446 (2016)
28. Zhu, M., Wang, X., Tarko, A.: Modeling car-following behavior on urban expressways in Shanghai: A naturalistic driving study. Transport Res. Part C Emerging Technol. 93, 425–445 (2018)
29. He, Z., Liang, Z., Wei, G.: A simple nonparametric car-following model driven by field data. Transport Res. Part B Methodol. 80, 185–201 (2015)
30. Hongfei, J., Zhichi, J., Anning, N.: Develop a car-following model using data collected by five-wheel system. IEEE Proc. Intell. Transp. Syst. 1, 346–351 (2003)
31. Khodayari, A., et al.: A modified car-following model based on a neural network model of the human driver effects. IEEE Trans. Syst. Man Cybern.: Syst. 42(6), 1440–1449 (2012)
32. Sunehag, P., et al.: Value-decomposition networks for cooperative multi-agent learning based on team reward. In: AAMAS Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, Stockholm, Sweden, pp. 2085–2087 (2018)
33. Rashid, T., et al.: QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. arXiv:1803.11485 (2018)
34. Alexey, D., et al.: CARLA: An open urban driving simulator. arXiv:1711.03938 (2017)
35. Jarrett, K., et al.: What is the best multi-stage architecture for object recognition. J. Mach. Learn. Res. 30(2), 2146–2153 (2009)
36. Bourerau, Y.L., et al.: Ask the locals: Multi-way local pooling for image recognition. In: Proceedings / IEEE International Conference on Computer Vision. IEEE International Conference on Computer Vision (IEEE ICCV), Barcelona, Spain, pp. 2651–2658 (2011)
37. Sutton, R.S., Barto, A.: Reinforcement Learning: An Introduction. 2nd ed. MIT Press, Massachusetts (2018)
38. Mnih, V., et al.: Playing atari with deep reinforcement learning. arXiv:1312.5602 (2013)
39. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature 518(7540), 329–335 (2015)
40. Van, H.H., Guez, A., Silver, D: Deep reinforcement learning with double q-learning. arXiv:1509.06461 (2015)
41. Wang, Z., et al.: Dueling network architectures for deep reinforcement learning. In: ICML, New York pp. 1995–2003 (2015)
42. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multi-armed bandit problem. Mach. Learn. 47(2-3), 235–256 (2002)
43. Wang, D., Deng, H., Pan, Z.: MRCDRL: Multi-robot coordination with deep reinforcement learning. Neurocomputing 406, 68–76 (2020)

How to cite this article: Yuan H, Ni J, Hu J. A centralised training algorithm with D3QN for scalable regular unmanned ground vehicle formation maintenance. IET Intell Transp Syst. 2021;15:562–572. https://doi.org/10.1049/itr2.12046