Research on Obstacle Avoidance Decision-Making for Unmanned Vehicles based on Reinforcement Learning

Hanfeng Xue
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, China
E-mail: 1817532845@qq.com

Zhenjie Zeng
School of Information Engineering
Xi'an Mingde University of Technology
Xi'an, China
E-mail: 937058108@qq.com

Pingping Liu
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, China
E-mail: 1341369601@qq.com

***Abstract*—In response to the limitations of traditional obstacle avoidance algorithms for unmanned vehicles in dealing with unknown obstacles and complex dynamic environments, this paper proposes Q Mixing Network and Video Delivery Network reinforcement learning algorithms, specifically for the research of obstacle avoidance decision-making for unmanned vehicles. By constructing a mapping relationship between local function values and global function values, it is possible to guide obstacle avoidance decisions for unmanned vehicles based on the decomposed function values. Experimental verification was conducted on a simulation platform based on ROS+Gazebo, and compared with the Quantile Regression for Reinforcement Learning algorithm in the same testing environment. The results showed that QMIX and VDN algorithms were more adaptable to complex map environments during training, with obstacle avoidance success rates increased by 16.9% and 18.1%, respectively, effectively improving the obstacle perception and avoidance capabilities of unmanned vehicles.**

Keywords-Reinforcement Learning; Unmanned Vehicle; Obstacle Avoidance Decision-Making; Autonomous Driving

I. INTRODUCTION

With the advent of the intelligent era, people's lifestyles have gradually become more convenient. In 2010, in order to prevent traffic accidents, the basic use mode of cars was changed, and unmanned cars, namely autonomous vehicle, emerged[1][2]. Autonomous vehicles are not only widely used in the transportation industry, but also in various fields such as automated logistics transportation, park patrols, and intelligent agriculture[3][4]. However, existing obstacle avoidance algorithms still lack emergency handling capabilities, which affects the safety of unmanned vehicles[5][6]. Therefore, in complex and unknown obstacle environments, how unmanned vehicles can effectively avoid obstacles and ensure smooth operation has become a key issue that urgently needs to be addressed.

Regarding obstacle avoidance for autonomous vehicles, existing research primarily falls into two categories: traditional planning algorithms and learning-based algorithms. Traditional methods such as A* algorithm, ant colony optimization (ACO), and dynamic window algorithm (DWA) demonstrate mature performance in static path planning but exhibit significant limitations when confronting complex dynamic environments. For instance, A* algorithm exhibits exponential computational complexity in intricate maps, failing to meet real-time requirements; DWA often gets stuck in local optima in densely obstructed scenarios, unable to generate optimal paths. Furthermore, these algorithms typically rely on precise prior maps, lacking adaptability in unknown environments and struggling with path optimization in continuous action spaces.

With the advancement of deep learning, end-to-end obstacle avoidance methods have emerged as a hot research topic[7]. Although deep neural networks enhance feature extraction capabilities, they often require massive labeled datasets, involve lengthy training cycles, and suffer from limited real-time responsiveness. In contrast, reinforcement learning (RL) optimizes policies through trial-and-error interactions between agents and environments[8], making it better suited for complex sequential decision-making problems. However, traditional RL algorithms exhibit computational inefficiency and struggle to capture long-term dependencies when handling multi-agent collaboration or high-dimensional state spaces[9] leading to response delays in emergency obstacle avoidance scenarios.

To overcome the limitations of both traditional and existing RL algorithms in complex dynamic environments, this paper proposes a reinforcement learning (RL) based obstacle avoidance decision method for autonomous vehicles. By establishing a mapping relationship between local and global Q-values and incorporating LSTM to enhance temporal feature extraction, this approach effectively addresses the curse of dimensionality in high-dimensional states. This paper compares the performance of QTRAN, QMIX, and VDN algorithms on the ROS+Gazebo simulation platform to validate the proposed algorithm's advantages in enhancing the flexibility and safety of autonomous vehicle obstacle avoidance.

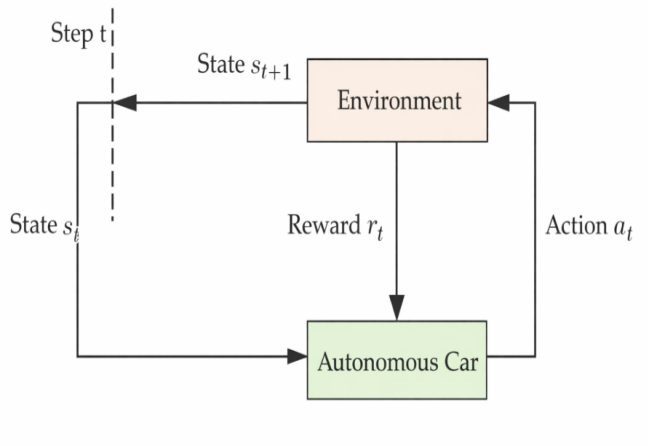


Figure 1. Reinforcement learning decision-making process

II. Algorithm Principle

A. Obstacle avoidance algorithm for unmanned vehicles based on QTRAN

The goal of quantile regression for reinforcement learning (QTRAN) is usually to optimize the Q-value function so that the agent can adopt the optimal behavioral strategy in a specific environment, thereby maximizing long-term returns[10]. The Q-value function essentially represents that when taking a certain action in a certain state, the agent continuously perceives the environmental state, selects the appropriate action, and adjusts the strategy based on the reward signal, ultimately achieving obstacle avoidance for unmanned vehicles.

The traditional Q-Learning method reflects the reward values of different states and actions by directly updating the Q-value table[11], but in complex environments, changes in Q-values are often not only dependent on the current state and action, but also influenced by historical behavior. Therefore, the QTRAN algorithm introduces a concept of "trajectory decomposition", which not only focuses on Q-value estimation under a single state, but also considers the global influence of the agent's behavior trajectory, making the Q-value estimation more refined. However, applying the QTRAN algorithm to the practical application of obstacle avoidance in unmanned vehicles may result in some shortcomings.

Firstly, obstacle avoidance tasks for unmanned vehicles typically require high levels of dynamism and real-time performance. In complex environments, autonomous vehicles must quickly respond to changes in the surrounding environment, such as the sudden appearance of obstacles or the behavior of other road users. The trajectory modeling method of QTRAN algorithm may result in lower computational efficiency of the algorithm. Especially in complex environments, QTRAN needs to comprehensively consider the impact of historical behavior in multi-step decision-making. This computationally intensive process may lead to reaction delays and cannot meet the high standards of real-time requirements for unmanned vehicles.

Furthermore, the QTRAN algorithm has high requirements for state and action space. In the obstacle avoidance task of unmanned vehicles, the state space and action space can be very large and complex, especially in dynamic and highly uncertain environments. How to effectively represent the state and select actions becomes a challenge. The QTRAN algorithm may face a curse of dimensionality when dealing with high-dimensional state spaces, causing the learning process to become slow and even unable to converge. In obstacle avoidance for autonomous vehicles, agents typically need to complete learning and make decisions in a very short amount of time, and the computational burden of QTRAN may cause the learning process to lag far behind real-time requirements.

Therefore, in response to the poor dynamic and real-time performance, as well as slow learning process of the QTRAN algorithm, this paper proposes a flexible, non-linear Q-value mixing and fast convergence QMIX and VDN, which can help unmanned vehicles make more flexible decisions to ensure efficient avoidance of obstacles.

B. Obstacle avoidance algorithm for unmanned vehicles based on QTRAN

In this article, QMIX approximates the Q-value function by training a neural network based on information such as the position, velocity, and distance of obstacles. Each time a decision is made, the algorithm selects the action with the highest Q value, such as turning, accelerating, decelerating, etc., to avoid obstacles and move towards the target. The process of updating the Q-value depends on a reward function, which rewards the vehicle based on its behavior. If the obstacle is successfully avoided, a positive reward is given; If there is a collision or deviation from the target, a negative reward will be given. By constantly interacting with the environment, the algorithm can gradually learn the optimal strategy for obstacle avoidance.

QMIX adopts a distributed strategy by using a nonlinear hybrid network to ensure monotonicity, and the same results can be obtained for the joint action value function Q_{tot} and the value function Q_a of a single agent[12].

$$Q_{tot}(\tau,\mu)=\sum_{t=1}^{n}Q_i(\tau_i,\mu_i;\theta_i) \quad (1)$$

Among them, τ_i is the observation sequence of intelligent agent i, which μ_i is the joint action of intelligent agent i.

In order to ensure the monotonicity of the value function to make formula (1) hold, a constraint is required between a joint action value function Q_{tot} and a single agent action value function Q_a :

$$\frac{\partial Q_{tot}}{\partial Q_a}\geq 0, \quad \forall a\in A \quad (2)$$

C. Obstacle avoidance algorithm for unmanned vehicles based on VDN

The VDN algorithm evaluates how to avoid each obstacle by assigning a local Q-value to it. In a multi obstacle environment, VDN decomposes the decision-making process and gradually optimizes each step of the decision. This not only improves the efficiency of convergence, but also effectively avoids collisions. Autonomous vehicles analyze each local Q value and integrate them to select the most suitable obstacle avoidance action for the current situation.

VDN adopts a backpropagation method to decompose the global value function into individual agents and linearly sum them, achieving good performance under centralized training[13]. Its joint action value function Q_{tot} is obtained by accumulating the value functions of each agent:

$$Q((o_1,o_2,\cdots,o_d),(a_1,a_2,\cdots,a_d))\approx\sum_{i=1}^{b}Q_I(o_i,a_i) \quad (3)$$

Among them, b represents b intelligent agents, o_i is the historical sequence of intelligent agent i, a_i is the selected action of intelligent agent, and Q_i is obtained from partial observation information of each intelligent agent.

III. Network Architecture Design

A. QTRAN algorithm

The QTRAN algorithm decomposes the global Q-value into the local Q-values of each agent and a transformation function as the network structure. In order to make the local Q-function values of the unmanned vehicle depend on its own local actions, QTRAN introduces an additional transformation process:

$$Q(\tau,\ \mu)=\sum_{i=1}^{N}Q_i(\tau_i,\mu_i)+\Delta_i(\tau,\mu) \tag{4}$$

Among them, $\Delta_i(\tau,\mu)$ is an additional term learned by the QTRAN network for adjusting local Q values.

For a single intelligent agent to have an action value function, the following relationship must be satisfied:

$$\arg\max Q_{jt}(\tau,\mu)=\begin{pmatrix}\arg\max_{\mu 1} Q_1(\tau_1,\mu_1)\\ \vdots \\ \arg\max_{\mu n} Q_n(\tau_n,\mu_n)\end{pmatrix} \tag{5}$$

During the training process, the autonomous vehicle calculates the expected return Q value $(Q_1(\tau_1,\mu_1),\ Q_2(\tau_2,\mu_2),\cdots,\ Q_N(\tau_N,\mu_N))$ for taking a specific action in a specific state based on the observed environmental state $(\tau_1,\ \tau_2,\cdots,\ \tau_N)$ and the action $(\mu_1,\ \mu_2,\cdots,\ \mu_N)$ taken. Calculate the loss of joint action value by comparing the joint action value of unmanned vehicles with the target value.

B. QMIX algorithm

QMIX uses a nonlinear mixture to calculate the global Q-value function, and introduces a neural network to calculate the local Q-value function for unmanned vehicles:

$$Q(\tau,\ \mu)=\sum_{i=1}^{N}\omega_i\left(Q_i(\tau_i,\mu_i)\right) \tag{6}$$

Among them, ω_i is the weight learned from a nonlinear function.

QMIX learns the strategy of unmanned vehicles by minimizing the loss function of the joint action Q function end-to-end[14].

$$L(\theta)=\sum_{i=1}^{b}\left[\left(r_{tot}+\gamma\max_{\mu'}Q_{tot}(\tau,\mu;\theta^-)-Q_{tot}(\tau,\mu;\theta)\right)^2\right] \tag{7}$$

Among them, b represents the number of samples sampled, r_{tot} is the actual reward value of all agents, θ^- is the target parameter, and $Q_{tot}(\tau,\mu;\theta^-)$ is the target network.

The architecture of the entire QMIX algorithm is shown in Figure 4. In Figure 4 (a), the expected return of a specific action taken by the unmanned vehicle in a given state based on its Q-value function is calculated by weighting the Q-value function with weights ω_i to calculate the expected return of the joint action of the unmanned vehicle. Part (b) of Figure 4 integrates the hybrid network based on the Q-value function and state of the unmanned vehicle, reflecting the expected return of the joint action of the unmanned vehicle. Part (c) of Figure 4 provides an in-depth display of the internal structure of an unmanned vehicle, including a Multilayer Perceptron (MLP) and a Gated Re current Unit (GRU). Observation of MLP processing agents o_t^i and previous actions μ_{t-1}^i extract features and provide a basis for decision-making. At the same time, the GRU layer processes time-series data to capture long-term dependencies of environmental states and outputs hidden states h_t^a。 Subsequently, the output h_t^a of GRU is combined with the extracted features of MLP to select action a_t through a network with an epsilon greedy strategy. In network design, specific activation functions are used for different functional modules. Within the GRU hidden layer, when encountering obstacles, the Sigmoid activation function is used, with an output range of [0,1]. The output range of Tanh activation function is [-1,1], with a mean close to 0, which enables the unmanned vehicle to distinguish the temporal state changes of acceleration and deceleration, thereby enhancing the modeling ability of motion trends. The fully connected layer uses ReLU activation function, mainly used to process sparse sensor data

and improve network training efficiency. This design significantly improves the decision robustness of obstacle avoidance scenarios through the scene adaptation of activation functions and the fusion mechanism of spatiotemporal features.

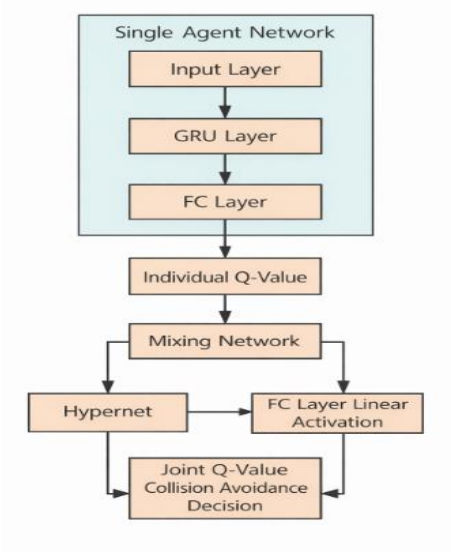


Figure 2. QMIX Independent Network Architecture Diagram.

C. VDN algorithm

The network structure of the VDN algorithm mainly relies on the sum of global Q values and local Q values to calculate the Q value of unmanned vehicles. Unmanned vehicles have their own observation spaces Obs1 and Obs2, and collect the speed and road conditions of the surrounding environment through sensors[15]. The decision-making process of autonomous vehicles is implemented by a neural network, which includes linear layers Linear and ReLU activation functions, as well as a long short-term memory network LSTM layer. The LSTM unit receives the observation space of the current time step and the hidden state generated by the LSTM layer of the previous time step. The input gate, forget gate, and output gate within the LSTM layer are filtered, updated, and integrated to ultimately output a new hidden state that contains all the temporal information required to understand the current scene. Specifically, LSTM can accurately learn whether obstacles and unmanned vehicles are accelerating, decelerating, or moving at a constant speed by continuously analyzing the velocity components in the observation space. Based on the motion trend and historical trajectory information encoded in this hidden state, the fully connected layer located after the LSTM can more accurately predict the possible location distribution of obstacles in the short term in the future. Furthermore, by integrating these predicted results with the planned trajectory of the vehicle, the system can more accurately determine the correct decision to avoid obstacles.

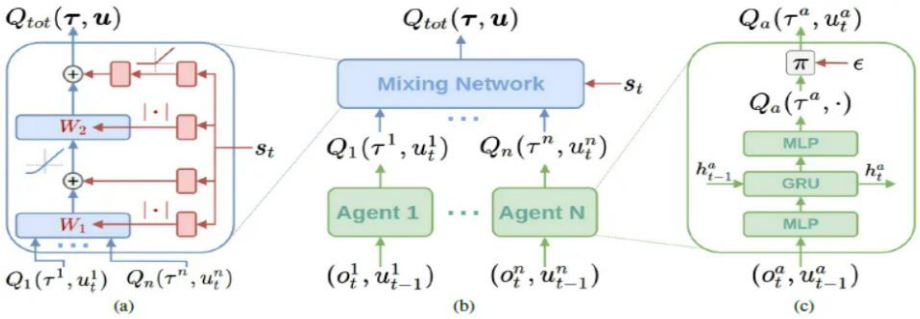


Figure 3. Structure diagram of the QMIX algorithm. (a) Hybrid network architecture. It is a fully connected network. (b) The overall architecture of QMlX. (c) Proxy network structure. For a single intelligent agent, it is necessary to learn an independent value function.

TABLE I. LSTM PARAMETER CONFIGURATION

Parameter	Value
LSTM hidden layer dimension	128
Time window length	10
Input feature dimension	20
time step	0.2s

At each time step, the autonomous vehicle calculates the state values V1 and V2 and the advantage values ADV1 and ADV2 based on its own observations. The state value represents the expected return in a specific state, while the advantage value measures the additional return of taking a certain action relative to other actions in a specific state. These values together constitute the

Q-values 1 and 2 of the autonomous vehicle, where the Q-value is the total reward for evaluating the specific actions taken in a given state.

The autonomous vehicle determines its behavior 1 and 1 by selecting the action that maximizes the Q value. This strategy allows the agent to choose the action with the highest expected return while considering the current environmental state and possible actions. Under obstacle avoidance by autonomous vehicles, this means that they will choose the path that is most likely to avoid collisions and safely reach their destination.

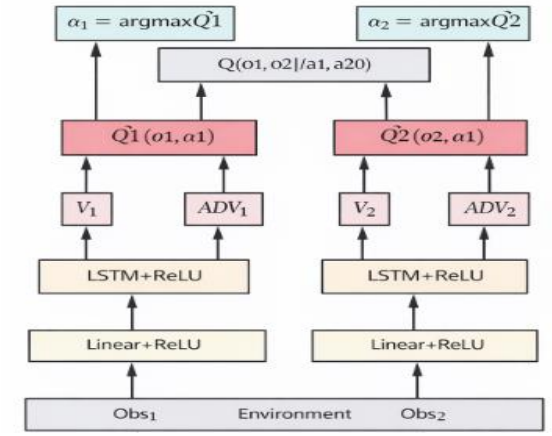


Figure 4. VDN independent intelligent agent network.

In order to verify the impact of temporal modeling capability on obstacle avoidance decision-making of unmanned vehicles, this study designed two sets of ablation experiments for comparative analysis. The experiment focuses on comparing two methods: the baseline method uses a fully connected network as the local Q-network of the agent, which only processes observations at the current time; The improvement method replaces the local Q-network with a structure of LSTM and fully connected network. In the experiment, the relevant parameters, training steps, and environment settings of the two algorithms were completely consistent, with only the local Q-network structure being changed. The experimental results show that, as shown in Figure 6, in the first 50 training iterations, both VDN base and VDN-LSTM algorithms had a win rate of 0, indicating that both algorithms failed to effectively avoid obstacles in the initial stage. As the training progresses, both algorithms show performance improvement. When the number of training iterations reaches 110, the win rate of VDN base algorithm reaches 100% and enters a stable state. In contrast, VDN-LSTM has higher learning efficiency, reaching its maximum win rate and entering a stable state after 80 iterations. By comparing the maximum win rate when the two reach a stable state, it can be found that the VDN-LSTM algorithm has achieved significant performance improvement compared to the VDN base algorithm, with a learning efficiency increase of 11.9%. This result clearly demonstrates that introducing the LSTM module into the VDN algorithm framework has a positive effect on improving the success rate of obstacle avoidance tasks.

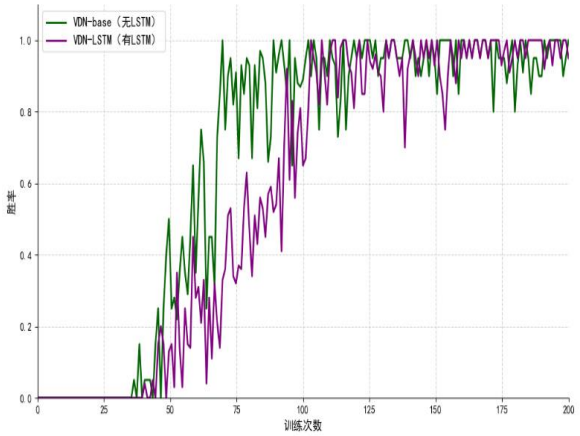


Figure 5. Comparison chart of ablation experiment performance.

IV. Algorithm Adaptive Optimization

A. GRU layer design

In the obstacle avoidance of unmanned vehicles, a single-layer GRU network is used as the core architecture. The input layer of the network receives feature vectors composed of the current frame LiDAR point cloud dimension, position and velocity state dimension, and previous time action dimension. By using a single-layer GRU unit for timing processing, the update gate dynamically adjusts the proportion of historical hidden states flowing into the current state, achieving selective retention and forgetting of historical information; The reset gate is responsible for filtering the relevance of the current input features and ultimately outputting a 128 dimensional hidden state h-t containing the temporal context of obstacle avoidance decisions. The hidden state is

then mapped to a 4-dimensional Q-value output through a fully connected layer. This design fully utilizes the adaptive characteristics of GRU gating mechanism, enabling it to dynamically adjust the dependency weights on historical memory based on the current input and scene, thereby efficiently capturing and processing key temporal dependencies in dynamic environments while maintaining low computational complexity. The specific GRU structure design is shown in Figure 7.

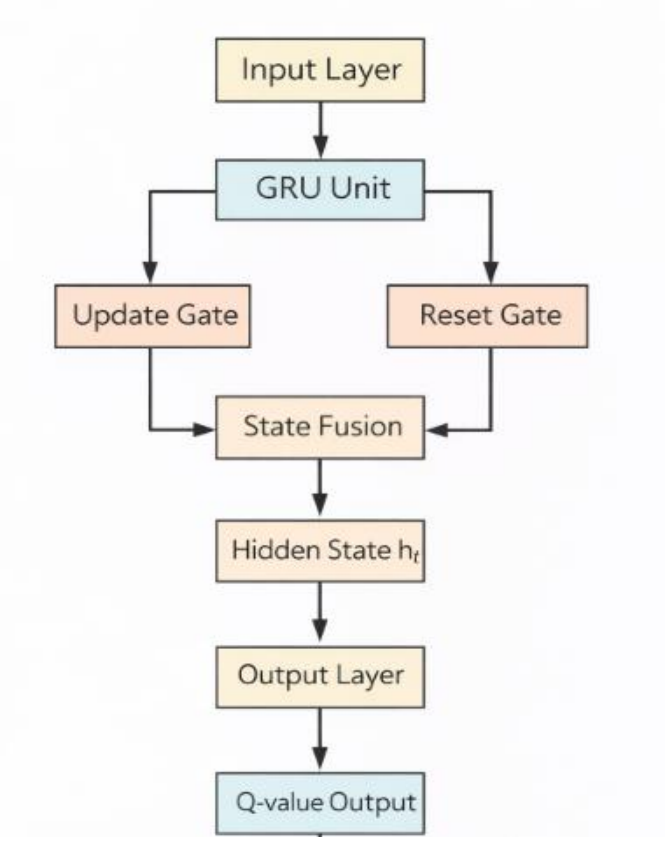


Figure 6. GRU structure design.

B. Action model adjustment

Unmanned vehicles predict particle pose updates by decomposing motion into a "rotation translation re rotation" process, and combine Gaussian noise models to reflect motion errors. The core parameters are shown in Table 2. Odom_alpha1 to Odom_alpha4 respectively control the uncertainty of translation and rotation: alpha1 and alpha2 quantify the translation error caused by geometric constraints during robot rotation, while alpha3 and alpha4 represent the drift of rotation angle during translation. For each particle, the new pose $x' = (x, y, \theta)$ is calculated based on the old pose $x = (x_0, y_0, \theta_0)$ and odometer measurement value $\delta = (\delta_{trans}, \delta_{rot1}, \delta_{rot2})$:

$$x' = x_0 + \delta_{trans} \cdot \cos(\theta_0 + \delta_{rot1}) \quad (8)$$

$$y' = y_0 + \delta_{trans} \cdot \sin(\theta_0 + \delta_{rot1}) \quad (9)$$

$$\theta' = \theta_0 + \delta_{rot1} + \delta_{rot2} \quad (10)$$

TABLE II. PARAMETER CONFIGURATION OF MOTION MODEL

Parameter	Description
odom_alpha1	Rotation component of translational noise
odom_alpha2	Translation component of translation noise
odom_alpha3	The rotational component of rotational noise
odom_alpha4	Translation component of rotational noise

C. Action model adjustment

In the obstacle avoidance scenario of unmanned vehicles, a reward function with a three-layer structure was designed to effectively guide the learning of efficient obstacle avoidance navigation strategies while ensuring safety, as shown in Table 3. The core of the security layer lies in applying significant negative incentives to collision behavior, setting the collision penalty to -10, in order to clearly highlight the serious consequences of collisions and encourage agents to prioritize avoiding collision risks in any situation. The efficiency layer aims to optimize the path selection and time efficiency in the obstacle avoidance process, and its design includes two key elements: one is to provide a positive+3 dynamic distance reward when the unmanned vehicle moves towards the target point and shortens the distance; The second is to impose a small -0.05 time step penalty to encourage faster navigation. The design of this layer ensures that, under the premise of successful obstacle avoidance, the cumulative reward of the unmanned vehicle on the path is always greater than zero, guiding it to choose a better path. The target layer is used for the completion of the final navigation task, and a

reward of +15 is given when the target point is successfully reached. At the same time, in order to accurately determine the arrival status, a threshold area with a radius of 0.5 around the target point is defined. Once the unmanned vehicle enters this area, it is considered to have successfully arrived. The reward value of the target layer is significantly higher than the punishment of the security layer and the possible small consumption of the efficiency layer, ensuring that the final total reward value obtained by successfully completing the entire navigation task is greater than 10, thereby strongly motivating the unmanned vehicle to safely and effectively reach the target point.

TABLE III. REWARD FUNCTION PARAMETER CONFIGURATION

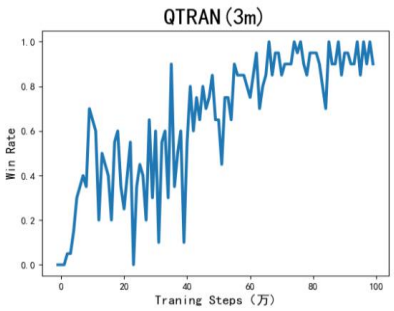
Parameter	Value
Collision penalty	-10
Target reward	20
progress coefficient	1
Minimum obstacle avoidance reward	3
Time penalty coefficient	-0.05

V. EXPERIMENTAL RESEARCH AND SIMULATION

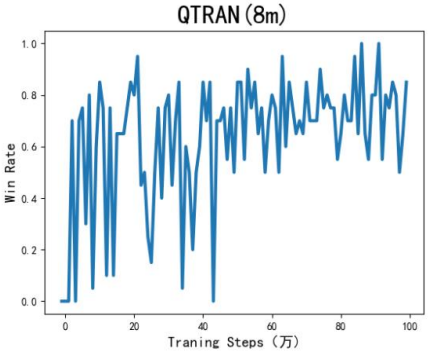
A. Experimental study

This article uses the StarCraft Multi Agent Challenge(SMAC) environment composed of PyCharm and StarCraftII for reinforcement learning training. Choose 3m.SC2Map, 8m.SC2Map, and 2s3z.SC2Map three maps were used to train QTRAN, QMIX, and VDN reinforcement learning algorithms respectively.

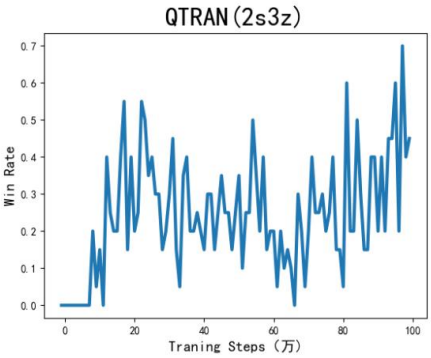
The evaluation of reinforcement learning algorithms in this article is based on the frequency of successful training for three different maps with the same number of training iterations and in the same virtual environment SMAC. In order to clearly demonstrate the win rate achieved by the three algorithms, curve graphs were drawn after the training of the three maps. The curve represents the trend of the frequency of successful interaction between the intelligent agent and the map environment during the training process.



(a)winning rate curve at 3m on map



(b)winning rate curve at 8m on map

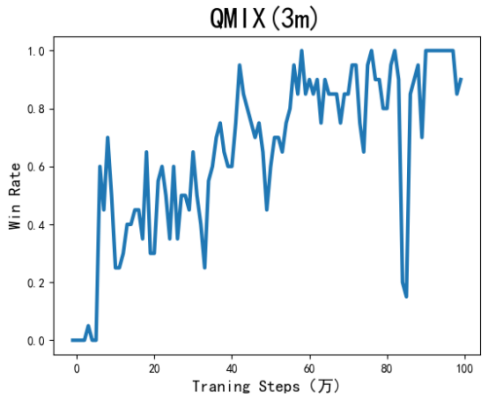


(c)winning rate curve at 2s3z on map

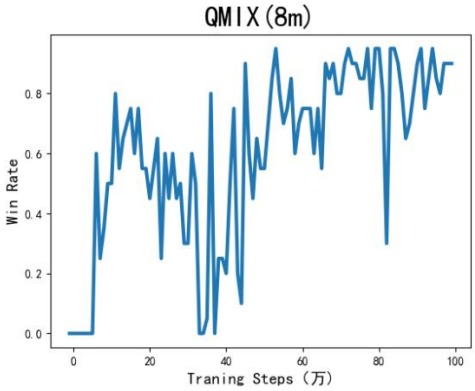
Figure 7. QTRAN algorithm's win rate curves.

After the training of QTRAN algorithm, the win rate curve is shown in Figure 5. Compared with the other two maps, the training success rate of 3m map is higher and it will reach stability earlier. Therefore, this algorithm performs well for 3m map. However, due to the high complexity of the 8m and 2s3z environments, the algorithm has a low frequency of interaction between these two maps. On the 8m map, the success rate only reaches 100% a few times, and finally stabilizes at around 80%. On the 2s3z map, with this number of training iterations, the success rate has not

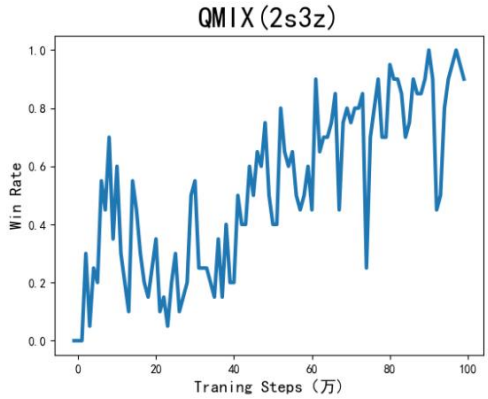
reached 100%, nor has it reached a stable value. It is still very unstable during early training and the effect is not very good. Secondly, it indicates that the fewer obstacles on the map, the higher the success rate of obstacle avoidance under the same number of training iterations. It also suggests that in order to achieve a higher success rate of obstacle avoidance, continuous training or the use of more effective algorithms is necessary.



(a)winning rate curve at 3m on map



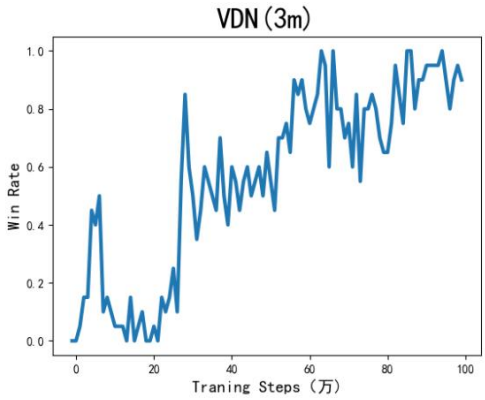
(b)winning rate curve at 8m on map



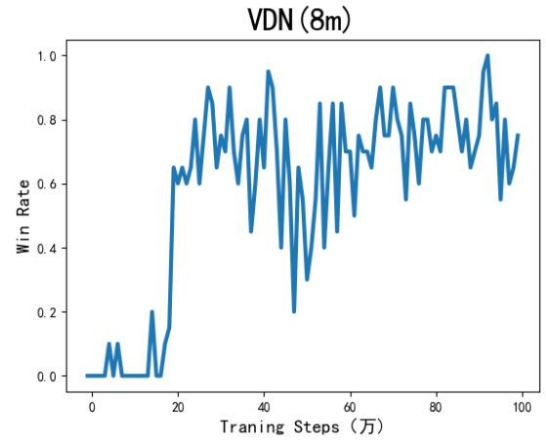
(c)winning rate curve at 2s3z on map

Figure 8. QMIX algorithm's win rate curves on different maps.

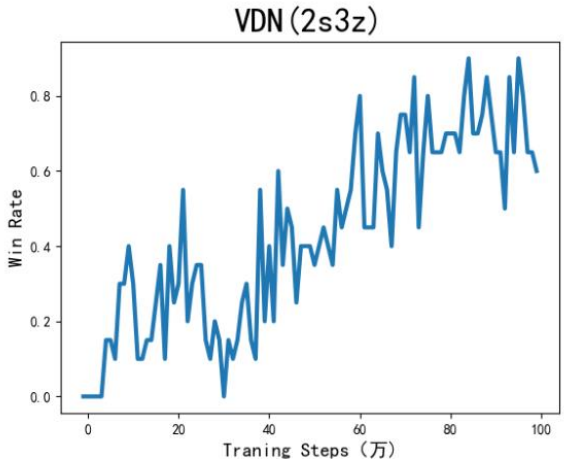
After the QMIX algorithm training is completed, as shown in Figure 6, the training success rate of the 3m map is higher compared to the other two maps, and it will reach stability earlier. Therefore, this algorithm performs well for 3m maps. However, due to changes in the local environment of 8m and 2s3z or issues with the algorithm itself, the frequency of interaction between these two maps is relatively low. In the 3m map, a stable 100% success rate is achieved first. However, in the 8m map, there was a trend of decreasing success rate in the middle stage. After the training ended, the success rate did not reach 100%, nor did it reach a stable value. In the 2s3z map, there was no significant decrease in success rate, and eventually there was a situation where the success rate was 100%, but it did not reach stability. Therefore, these two maps still need a long time of training to reach a stable high success rate value. Secondly, it also indicates that the fewer obstacles on the map, the higher the success rate of obstacle avoidance under the same number of training iterations. It also suggests that in order to achieve a higher success rate of obstacle avoidance, continuous training or the use of more effective algorithms is necessary.



(a)winning rate curve at 3m on map



(b)winning rate curve at 8m on map



(c)winning rate curve at 2s3z on map

Figure 9. VDN algorithm's win rate curves on different maps.

After the VDN algorithm training is completed, as shown in Figure 7, the training success rate of the 3m map is higher compared to the other two maps, and it will reach stability earlier. Therefore, this algorithm performs well for 3m maps. However, due to the high environmental complexity of 8m and 2s3z, the frequency of interaction between the two maps using this algorithm is relatively low. In the 3m map, the success rate appeared and stabilized at 100% during the later training process. In the 8m map, the success rate almost did not reach 100%. Only by increasing the number of training times can the success rate be stabilized at a higher value. In the 2s3z map, there was no 100% success rate, so it is also necessary to increase the number of training times to make the success rate close to the higher value. Secondly, it also indicates that the fewer obstacles on the map, the higher the success rate of evasion under the same number of training iterations. It also suggests that in order to achieve a higher success rate of obstacle avoidance, continuous training or the use of better performing algorithms is necessary, demonstrating that the algorithm performs very well in simulation environments.

B. Experimental analysis and comparison

In order to compare the efficiency of QTRAN, QMIX, and VDN algorithms in successfully interacting between intelligent agents and environmental maps on 3m, 8m, and 2s3z maps, curve graphs were drawn on the three maps.

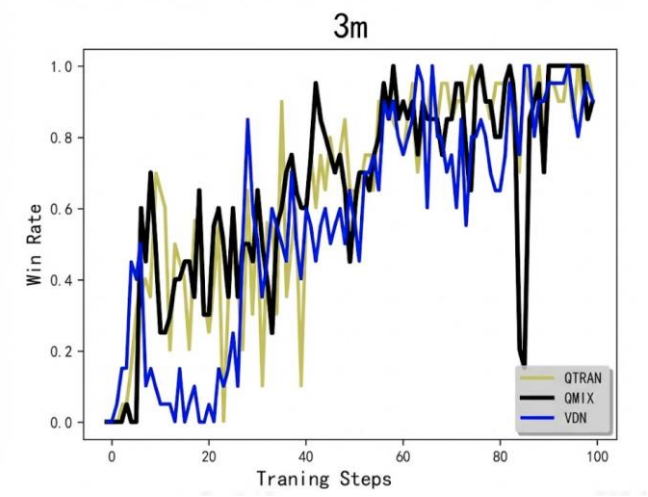


Figure 10. Analysis curve chart under 3m map.

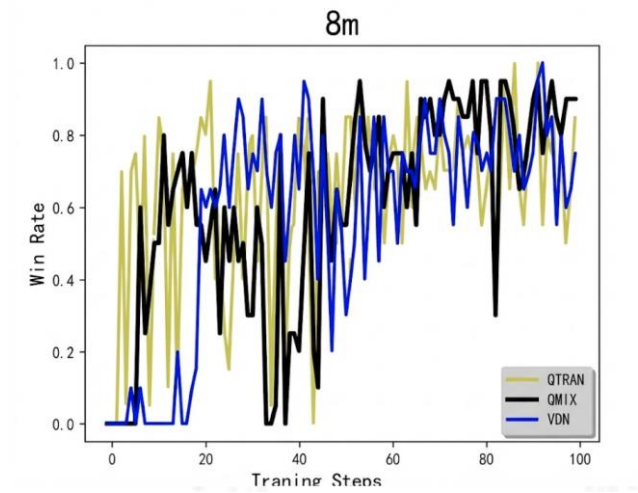


Figure 11. Analysis curve chart under 8m map.

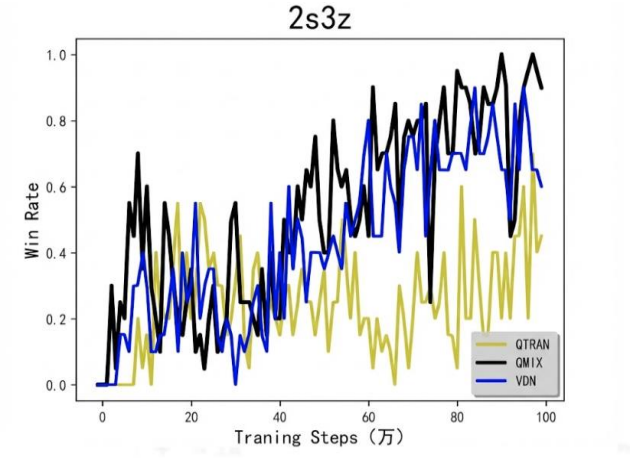


Figure 12. Analysis curve chart under 2s3z map.

Overall, the fewer obstacles present in the map, the easier it is for reinforcement learning algorithms to capture optimal strategies, thereby restoring previous success rates sooner. Conversely, as obstacle density increases, the training cycles required for algorithmic exploration and training become more demanding. Such high-difficulty environments serve to more clearly distinguish differences in efficiency and applicability among various algorithms. In simple environments with smaller state spaces and sparse obstacles (3m maps), the QTRAN algorithm demonstrates a certain advantage by executing its linear hierarchical strategy with faster convergence, achieving stable high success rates earlier than other algorithms. QMIX and VDN, though similarly present, ultimately achieve success rates close to 100%. However, in complex environments with increased obstacle density and enhanced dynamics (8m and 2s3z maps), the QTRAN algorithm's performance noticeably degrades, with its success rate barely maintaining around 80%. In contrast, QMIX and VDN algorithms demonstrate notable robustness in complex training scenarios. As training iterations increase, both algorithms converge to a higher and more stable success rate earlier than QTRAN. Quantitative data reveals that, at identical training steps, QMIX and VDN achieve average obstacle avoidance success rates 16.9% and 18.1% higher than QTRAN on complex maps. Notably, the VDN algorithm, benefiting from its mechanism of decomposing global rewards into local rewards and the introduction of LSTM-related modules, achieved a significant performance boost in the 2s3z scenario. This improvement is primarily driven by QMIX's nonlinear hybrid network and VDN's value concatenation mechanism, which more accurately elucidate the joint action value function in complex environments. Consequently, these algorithms guide autonomous vehicles to make optimal decisions in navigating narrow spaces and dynamic obstacles.

C. Obstacle avoidance simulation experiment

Robot Operating System (ROS)[16] is a software package that can be used to organize code and provide simulation environments, such as Gazebo[17]. Gzaebo supports various sensor models, engines, and control systems, and can simulate many intelligent agents, including drones, unmanned vehicles, and mobile robots. Therefore, this article uses ROS and Gazebo simulation environments to train mobile robots based on three algorithms, and tests them on different simulation maps. To enhance the universality and validation validity of experimental conclusions, Deep Q Network (DQN) was specifically introduced as a benchmark reference algorithm for rigorous comparative analysis with three algorithms. The specific parameter settings for the simulation environment are shown in Table 4.

TABLE IV. REWARD FUNCTION PARAMETER CONFIGURATION

Parameter	Value
Radar update frequency	5.5Hz
Radar bracket length	0.15m
Radar bracket radius	0.02m
maximum linear velocity	15m/s
acceleration	$-3\,m/s^2 \sim +3\,m/s^2$
Maximum detection distance	30m
Gaussian noise	0.01
resolution width	360px
Camera update frequency	30Hz
Maximum steering angular velocity	1.8rad/s
Minimum distance between obstacles	5m
Maximum output torque	30N·m

In order to quantitatively evaluate and visually compare the training efficiency and performance of DQN algorithm with the other three

reinforcement learning algorithms in the simulation environment, the trend of the reward values generated by each algorithm over time during the entire simulation training cycle is plotted as a learning curve graph. This curve graph can clearly demonstrate the convergence speed, learning stability, and ultimately achieved asymptotic performance level of each algorithm, providing a direct and visual basis for comparing the efficiency and effectiveness between algorithms.

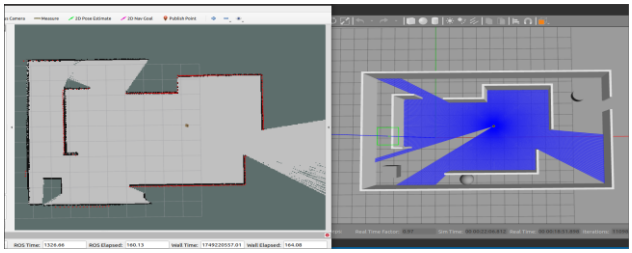


Figure 13. QTRAN algorithm training map.

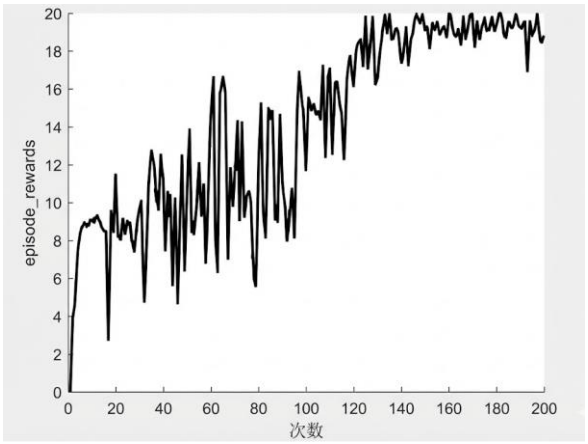


Figure 14. QTRAN algorithm training map.

The DQN algorithm, as a classic reinforcement learning algorithm, can be compared horizontally with these three algorithms. The training reward curve shown in Figure 15 indicates that during the algorithm initialization phase, due to sparse environmental rewards, the performance of unmanned vehicles fluctuates greatly, with average reward values fluctuating within the range of [5,15]. When the training cycle reaches about 120 times, the algorithm performance improves and the reward value can quickly exceed 10. However, the DQN algorithm is relatively inefficient in dealing with such complex environments, which results in the reward value not quickly converging to a higher and more stable level. In addition, the evaluation data shows that under this training state, the probability of the unmanned vehicle successfully avoiding obstacles is 72%.

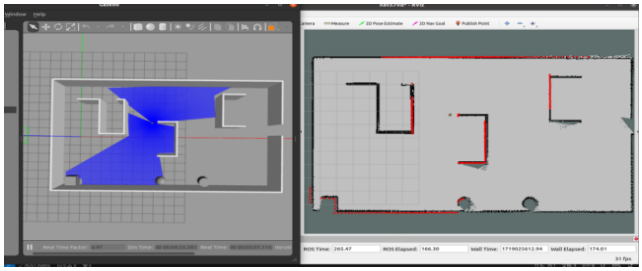


Figure 15. QTRAN algorithm training map.

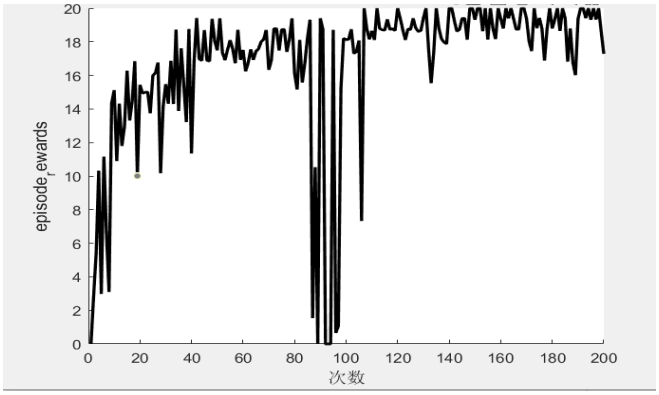


Figure 16. QTRAN algorithm training reward.

The map trained by QTRAN algorithm is shown in Figure 16, and the reward values generated are shown in Figure 17. The training times are 200. In the early stages of training, unmanned vehicles are not familiar with their environment and are still in an observation stage, so the reward value for learning is very low. At around 100 times, there was a sharp decrease in the reward value, which was due to the complex environment near the car or the low reward value caused by the reward mechanism of the QTRAN algorithm. According to the reward function settings, the reward value for collision is -10, the reward value for successfully avoiding obstacles is greater than 0, and the reward value for successfully avoiding obstacles and reaching the target point is greater than 10. When the number of training sessions exceeds 110, the reward value is generally greater than 10. In the end, the probability of successfully avoiding obstacles in 200 training sessions was 77%.

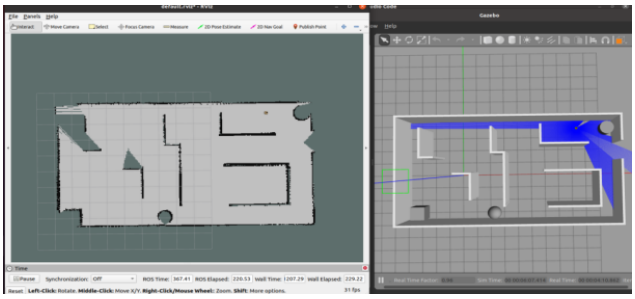


Figure 17. QMIX algorithm training map.

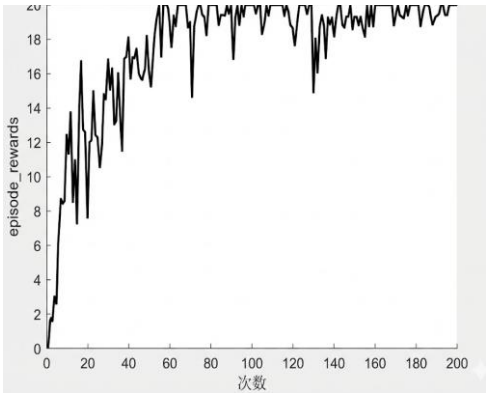


Figure 18. QMIX algorithm training reward.

As shown in Figure 19, the reward value curve of QMIX algorithm exhibits high stability and shows a continuous upward trend overall. Observing the initial stage of training based on Figure 18, due to the fact that the unmanned vehicle is still in the stage of environmental exploration and strategy learning, and is not familiar with the simulation environment, its reward value is relatively low. However, even at this stage, the reward value of QMIX has maintained a stable growth, indicating that its learning process is relatively smooth. After more than 40 training iterations, the performance of QMIX algorithm significantly improves, and its reward value remains stable at 10 or above. QMIX exhibits significant advantages in key indicators of obstacle avoidance performance: compared to QTRAN algorithm, its obstacle avoidance success rate has increased by 16.9% and convergence speed has increased by 50.9%; Compared to the DQN algorithm, its obstacle avoidance success rate has increased by 17.3%. These data strongly demonstrate the comprehensive superiority of QMIX algorithm in training stability, convergence speed, and final obstacle avoidance performance.

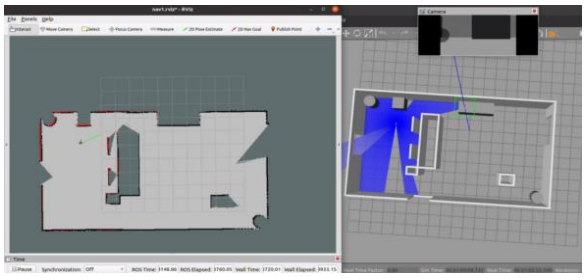


Figure 19. VDN algorithm training map.

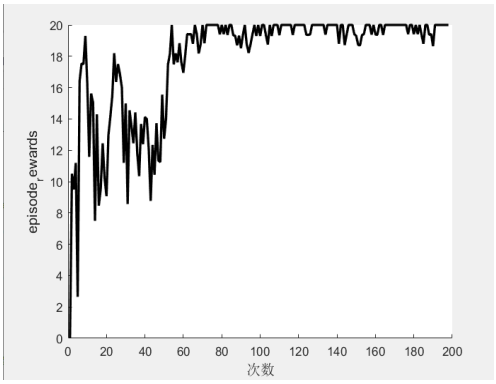


Figure 20. VDN algorithm training reward.

In the early stages of training, the VDN algorithm exhibits significant fluctuations in reward values, as shown in Figure 21. This phenomenon can be attributed to the use of sensors by unmanned vehicles for intensive environmental perception and exploration, as shown in Figure 20, resulting in unstable behavior strategies and significant fluctuations in reward feedback. However, when the training iterations exceeded 40 times, the VDN algorithm combined activation function and LSTM for training, and its performance rapidly improved and tended to stabilize, with the reward value consistently maintained above 10. In the comparison of key performance indicators, VDN exhibits significant advantages: compared to QTRAN algorithm, VDN's final average reward value has increased by 18.1% and convergence speed has increased by 51.8%; Compared to the DQN algorithm, the obstacle avoidance success rate of VDN has increased by 19.5%. These results indicate that although VDN experienced significant performance fluctuations in the early stages of exploration, it was able to quickly converge and ultimately achieve excellent performance levels by combining LSTM and learning mechanisms,

especially surpassing QTRAN and DQN algorithms in obstacle avoidance tasks.

VI. Conclusions

This article focuses on the limitations of traditional obstacle avoidance algorithms for unmanned vehicles in dealing with unknown obstacles and complex dynamic environments. A reinforcement learning based obstacle avoidance decision-making method for unmanned vehicles is established on maps of different levels of complexity. The reinforcement learning obstacle avoidance algorithm and simulation system for unmanned vehicle obstacle avoidance are used for research.

This study employs the QTRAN reinforcement learning algorithm, integrating factors such as the perceptual capabilities of unmanned vehicles and environmental complexity, to perform obstacle avoidance tests in a simulated environment. The simulation results demonstrate that the QTRAN algorithm exhibits inferior dynamic performance, inadequate real-time responsiveness, and a sluggish learning process. To tackle these drawbacks, the QMIX and VDN unmanned vehicle obstacle avoidance algorithms are put forward. In training experiments utilizing the QMIX and VDN algorithms, the success rates of obstacle avoidance are enhanced by 16.9% and 18.1% respectively. To address these issues, QMIX and VDN unmanned vehicle obstacle avoidance algorithms were proposed. In the training experiments using QMIX and VDN algorithms, the obstacle avoidance success rates were improved by 16.9% and 18.1%, respectively. When the training times reached about 60 times, the two algorithms basically converged, and the convergence speed increased by 50.9% and 51.8%, respectively. In terms of resource consumption, the algorithm reduced the memory usage by 39.5% and 44.9%, respectively. On this basis, it has been applied to the simulation environment of unmanned vehicles, improving their safety and efficiency.

When solving the obstacle avoidance decision-making problem of unmanned vehicles, both QMIX and VDN algorithms have significant advantages over QTRAN algorithm in terms of environmental complexity and convergence during algorithm training, effectively improving the obstacle perception and avoidance ability of unmanned vehicles. The QMIX and VDN algorithms achieved remarkable obstacle avoidance performance in simulated environments, but transferring them to real-world physical settings remains a significant challenge—a key focus for future research. First, real-world scenarios exhibit extreme complexity. Not only do weather conditions like rain, snow, and fog introduce dynamic variables, but sensor noise from LiDAR and cameras is far more intricate than in simulations. This complexity creates blind spots, necessitating further research into robust state estimation methods that integrate multiple sensors. Second, prediction uncertainty is inherently high. Unlike objects in simulations that follow predefined rules, pedestrians and non-operational entities on real roads exhibit strong randomness and social interaction attributes. Future work must incorporate early warning recognition modules to enrich the state space of reinforcement learning with predictions of surrounding traffic participants' behaviors. Finally, given the limited computational capacity of real truck computing platforms, while RL models based on LSTMs and reinforcement networks are expected to grow in complexity, the next step involves lightweight pruning or knowledge upgrading of obstacle models. This will ensure real-time obstacle avoidance control remains feasible in subsequent levels.

References

[1] Faisal A, Kamruzzaman M, Yigitcanla T, et al. Understanding autonomous vehicles[J]. Journal of transport and landuse, 2019, 12(1): 45-72.

[2] Bathla G, Bhadane K, Singh R K, et al. Autonomous vehicles and intelligent automation: Applications, challenges, and opportunities[J]. Mobile Information Systems, 2022, 2022(1): 7632892.

[3] Mohsan S A H, Khan M A, Noor F, et al. Towards the unmanned aerial vehicles (UAVs): A comprehensive review[J]. Drones, 2022, 6(6): 147.

[4] Gupta A, Afrin T, Scully E, et al. Advances of UAVs toward future transportation: The state-of-the-art, challenges, and opportunities[J]. Future transportation, 2021, 1(2): 326-350.

[5] Rezaee M R, Hamid N A W A, Hussin M, et al. Comprehensive review of drones collision avoidance schemes: Challenges and open issues[J]. IEEE Transactions on Intelligent Transportation Systems, 2024.

[6] Lyu M, Zhao Y, Huang C, et al. Unmanned aerial vehicles for search and rescue: A survey[J]. Remote Sensing, 2023, 15(13): 3266.

[7] Chib P S, Singh P. Recent advancements in end-to-end autonomous driving using deep learning: A survey[J]. IEEE Transactions on Intelligent Vehicles, 2023, 9(1): 103-118.

[8] Padakandla S. A survey of reinforcement learning algorithms for dynamically varying environments[J]. ACM Computing Surveys (CSUR), 2021, 54(6): 1-25.

[9] Liang J, Miao H, Li K, et al. A review of multi-agent reinforcement learning algorithms[J]. Electronics, 2025, 14(4): 820.

[10] PANG Jinhui, FENG Zicong.A review of exploration methods for deep reinforcement learning based on uncertainty[J]. Computer Application Research,2023,40(11):3201-3210.

[11] FANG Guoquan, ZHAO Jun, CHEN Hao, et al. Task planning method for multi inspection robot collaboration in substations[J]. Guangdong ElectricPower,2024,37(11):47-54.

[12] Rashid T, Samvelyan M, De Witt C S, et al. Monotonic value function factorisation for deep multi-agent reinforcement learn¬ing[J]. Journal of Machine Learning Re¬search, 2020, 21(178): 1-51.

[13] Sunehag P, Lever G, Gruslys A, et al. Value-decomposition networks for coop¬erative multi-agent learning[J]. arxiv pre¬print arxiv:1706.05296, 2017.

[14] Son K, Kim D, Kang W J, et al. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning[C]//International conference on machine learning. PMLR, 2019: 5887-5896.

[15] CAO Yingjie Research on the Deploy-ment Strategy of Emergency Unmanned Aerial Vehicle Base Stations Based on Deep Reinforcement Learning [D]. University of Electronic Science and Technol¬ogy of China, 2024.

[16] Robot Operating System (ROS)[M]. Cham, Switzerland: Springer, 2017.

[17] WU Fan Trajectory Planning Algorithm and Autonomous Driving Platform Based on Complex Optimization [D]. Dalian University of Technology, 2021.