Autonomous On-ramp Merge Strategy Using Deep Reinforcement Learning in Uncertain Highway Environment

Sifan Wu
1. School of Transportation Science and Engineering, Beihang
2. Beijing Key Laboratory for Cooperative Vehicle Infrastructure Systems&Safety Cooperative Control
3. Beijing Advanced Innovation Center for Big Data and Brain Computing
Beijing, China
wusifan9070@163.com

Daxin Tian*
1. School of Transportation Science and Engineering, Beihang
2. Beijing Key Laboratory for Cooperative Vehicle Infrastructure Systems&Safety Cooperative Control
3. Beijing Advanced Innovation Center for Big Data and Brain Computing
Beijing, China
dtian@buaa.edu.cn

Jianshan Zhou
1. School of Transportation Science and Engineering, Beihang
2. Beijing Key Laboratory for Cooperative Vehicle Infrastructure Systems&Safety Cooperative Control
3. Beijing Advanced Innovation Center for Big Data and Brain Computing
Beijing, China
jianshanzhou@foxmail.com

Xuting Duan
1. School of Transportation Science and Engineering, Beihang
2. Beijing Key Laboratory for Cooperative Vehicle Infrastructure Systems&Safety Cooperative Control
3. Beijing Advanced Innovation Center for Big Data and Brain Computing
Beijing, China
duanxuting@buaa.edu.cn

Zhengguo Sheng
1. Department of Engineering and Design
2. University of Sussex
Richmond, UK
z.sheng@sussex.ac.uk

Dezong Zhao
1. James Watt School of Engineering
2. University of Glasgow
Glasgow, UK
Dezong.Zhao@glasgow.ac.uk

***Abstract*—On-ramp merge is a complex traffic scenario in autonomous driving. Because of the uncertainty of the driving environment, most rule-based models cannot solve such a problem. In this study, we design a Deep Reinforcement Learning (DRL) method to solve the issue of ramp merges in uncertain scenarios and modify the structure of the Twin Delayed Deep Deterministic policy gradient algorithm (TD3), using Long Short-Term Memory (LSTM) to select an action based on temporal information. The proposed method is applied in the on-ramp merge and verified in the Simulation of Urban Mobility (SUMO). Results show that the proposed method performs significantly better generalization in uncertain traffic scenarios.**

Index Terms*—On-ramp merge, deep reinforcement learning (DRL), Long Short-Term Memory (LSTM), simulation of urban mobility (SUMO)

I. Introduction

Nowadays, advanced driver assistance systems (ADAS) have been widely applied in automotive systems and can effectively reduce the occurrence of accidents caused by drivers [1]. With successful application of ADAS with vehicles, people are more eager to have fully autonomous vehicles. However, there will be a long period time before fully autonomous driving appears. Nowadays, Some companies are working on new technologies to solve issues with autonomous driving and handle more complex tasks such as merging, lane change and uncontrolled intersections. Although vehicle-to-everything (V2X) can effectively ensure the safety of autonomous vehicles, large-scale deployment of V2X devices is still challenging [2]. Therefore, how to improve the performance of autonomous vehicles without V2X communications is very important.

For the AVs, the problem of ramp merge has always been difficult, and it is a key problem that needs to be solved urgently. Ramp merge mainly needs to consider the situation of the traffic flow on the main road in multi-vehicle traffic scenarios, and the surrounding environment at an intersection. How to make the behavior of autonomous vehicles similar to the behavior of the human driver, taking appropriate decisions has always been the focus of research [3].

At present, a considerable body of literature has been published on the ramp merge. Zhou et al. [4] applied the collaborative approach to the traditional IDM model, which

∗Corresponding Author

This research was supported in part by the National Natural Science Foundation of China under Grant No. U20A20155, and the Beijing Municipal Natural Science Foundation under Grant No. L191001.

can greatly improve the traffic capacity through the cooperative behavior of vehicle-to-vehicle. Pei et al. [5] optimized the trajectory of the merge according to the cost of the merging time and proposed a planning algorithm based on cooperation. The algorithm determines the opportunity to merge according to the cooperation between the vehicle and another vehicle and the timetable to ensure that the vehicle adopts a safe and effective merging behavior. Zhou et al. [6] applied the pontryagin maximum principle to solve the optimal control problem and used a recursive planning framework to predict the vehicle's trajectories. Experiments showed that the algorithm could solve the problem of specifying the merging point and performed well in moving the merging vehicles to the desired positions. However, the traditional ramp merge strategies are rule-based models, which do not have learnability and the model uses a fixed rule to deal with complex scenarios so that the behavior is relatively conservative.

Nowadays, DRL has been gradually applied in the field of transportation and it has been proven to solve various problems in traffic scenarios, such as ramp merge [7]. Isele et al. [8] proposed the Deep Q-Networks (DQN) to deal with the scenario of intersection merge. This method uses three different representations of ramp merge (time-to-go, sequential actions, and creep-and-go) to ramp merge, obtaining good results for different scenarios (turn left, turn right, go straight at the intersection, and change lanes) and achieving a high success rate. Wang et al. [9] proposed a DQN model to handle the problem of parallel-type on-ramp. However, the DQN model still uses a discrete action space to describe the merging behavior. Compared with the larger action space of the merging behavior, the use of discrete actions obviously cannot be well applied to the merging model. A discrete action space describes actions, which makes the action behavior relatively limited. Wang et al. [10] used a continuous action space to study the ramp merge, applied Long-Short Term Memory (LSTM) [11] to predict the initial state at each moment, and proposed a Q-function approximator for the selection of the optimal action in the given state. Lin et al. [12] used the Deep deterministic policy gradient (DDPG) to study the ramp merge and achieved the smallest collision and higher comfort by modifying the weight of the jerk penalty. However, DDPG algorithm overestimated the Q value in the actual process, which affected the performance of the algorithm and the learning efficiency. In our study, we proposed the twin delayed deep deterministic policy gradient (TD3) algorithm [13] to solve the problem of ramp merge in an uncertain environment. Compared with the DDPG, the TD3 improves the shortcomings of the overestimation of the Q value function and guarantes more stable training of actor network, which enhance the performance of the DDPG. The main contributions of this work are two-fold. Firstly, an recurrent neural network module (RNN) is added to the actor network in TD3 to process temporal information. LSTM is used as basic recurrent neural network module (RNN). To the best of our knowledge, there are no published papers using this structure for TD3. Secondly, the proposed method can effectively handle sensor noise and is suitable for a realistic traffic environment.

The remainder of this paper is organized as follows: Section II describes the preliminaries of reinforcement learning and introduces the proposed method. Section III formulates merging problems. The experiment are showed in Section IV. Finally, the conclusion and the discussion are presented in Section V.

II. Methodology

A. Reinforcement Learning

The object of reinforcement learning includes the agent and the environment in which it interacts. In this paper, the agent is the merging vehicle, and the environment is the traffic environment. The agent chooses the appropriate action based on the generated policy $\pi(s)$ and the environment updates the environment state s_{t+1} at the next step, and returns to single-step reward or punishment r_{t+1}, repeat this process until the task is completed or the termination. The agent's strategy for selecting an action is to maximize the expected (discount) cumulative rewards.

B. Deep Deterministic Policy Gradient (DDPG)

Deep deterministic policy gradient (DDPG) [14] is a deterministic policy-based reinforcement learning algorithm that uses the actor-critic framework. The actor network is a policy network composed of two networks: the eval network and the target network. The eval network is used to select the action according to the state, and the target network calculates the Q value in the critic target network. The update of the actor eval net adopts a deterministic policy gradient, as shown in (1).

$$\bigtriangledown J(\phi) = \frac{1}{N} \sum \nabla_a Q_\theta|_{a=\pi_\phi(s)} \nabla_\phi \, \pi_\phi(s). \tag{1}$$

The critic network also consists of two networks: the eval net and the target net. The target network updates the target Q value in the target net, as described in (2).

$$Q_{\theta^-} = r + not\_done * \gamma * Q_{\theta^-}(s', \pi_\phi(s')). \tag{2}$$

The critic network uses TD-error to update the eval network, as shown in (3).

$$TD_\delta = \frac{1}{N} \sum \left(Q_\theta - Q_{\theta^-}\right)^2. \tag{3}$$

For the target net, the soft update is used to update the parameters in the target net, as described in (4) and (5).

$$\theta^- \leftarrow \varepsilon\theta + (1-\varepsilon)\theta^-, \tag{4}$$

$$\phi^- \leftarrow \varepsilon\phi + (1-\varepsilon)\phi^-. \tag{5}$$

C. Twin Delayed Deep Deterministic Policy Gradient (TD3)

While DDPG has performed well in various fields, it still has some problems, such as unsmooth policy and overestimation. The TD3 algorithm addresses the issue of the above DDPG algorithm. Thus, we use TD3 to solve the problem of ramp merge. Compared with the DDPG algorithm, the TD3 algorithm has the following improvements:

- The TD3 uses two independent critic networks to calculate the target Q in network and take the minimum between the two estimates to avoid overestimation.

$$Q_{\theta^-} = r + not\_done * \gamma * \min_{i=1,2} Q_{\theta_i^-}(s', \pi_{\phi_1}(s')). \tag{6}$$

- In the learning process, as compared to the single-step update of the critic network, the actor network adopts delaying policy updates (update step: $d > 1$). By this modification, the training of the actor network is guaranteed to be more stable.
- When calculating the target action, noise based on normal distribution ($\varepsilon \sim N(0, \sigma)$) is used to reduce the estimation error of value.

$$Q_{\theta^-} = r + not\_done * \gamma * \min_{i=1,2} Q_{\theta_i^-}(s', \pi_{\phi_1}(s') + \text{clip}(\varepsilon, -c, c)). \tag{7}$$

D. The proposed method

In the realistic traffic environment, perceptual noise always occurs in sensors. To deal with sensor noise, a recurrent neural network module is added to the neural network for TD3. LSTM is applied to handle temporal sequence, we only replace the first layer of the actor network with LSTM to handle temporal information. This helps the vehicle choose an action based on temporal information rather than just current information. We refer to the improved TD3 algorithm as Recurrent based Twin Delayed Deep Deterministic Policy Gradient (RTD3). RTD3 can handle long-term input sequences and the prediction of temporal information, which helps the agent effectively solve the difficulties caused by partial observations in the environment.

Fig. 1 shows the structure of RTD3. An overall learning process is shown. The black line represents real-time updates($d = 1$). The blue line represents the delayed update($d > 1$). In the figure, the processor module is used to keep the last k states. c_1 and c_2 are the initial state of LSTM.

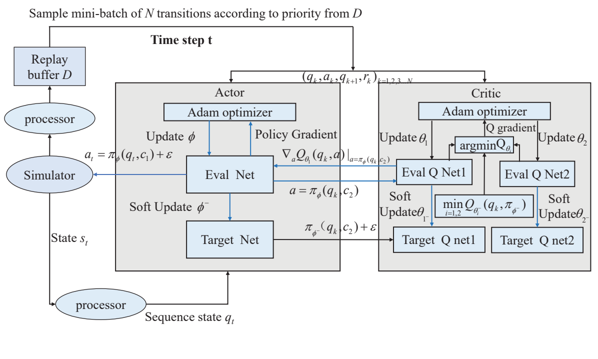

Fig. 1. Structure of the proposed method.

III. Merging Problem Formulation

A. Merging Environment

We build a simulation scenario using Simulation of Urban Mobility (SUMO) [15]. In the simulated scenario, simulated scenario is built on the basis of the real world of I-80, as shown in Fig. 2. The I-80 is a highway with an on-ramp with a length of approximately 500 m (1650 feet), where the on-ramp is approximately 125 m (420 feet) from the start of the highway and approximately 375 m (1230 feet) from the end of the highway. There is a buffer area of approximately 95 m (310 feet) from the entrance of the ramp to the main highway. The main highway has six main lanes, the width of lanes 1 to 5 is 3.75 m, the width of lane 6 at the ramp entrance is wider than 4.8 m.

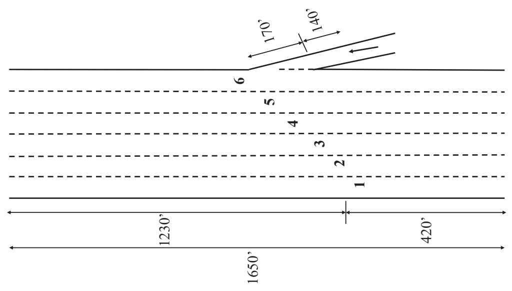

Fig. 2. Scenario of ramp merge.

As shown in Fig. 3, the RL algorithm is used to control the merging vehicle only in the control zone. When leaving or not entering the control zone, the merging vehicle is not controlled by the control algorithm. Control zones are set between 160 m before the merging point and 60 m after the merging point in the simulated scenario because when the vehicle ramped into the main road, it needed to achieve stability.

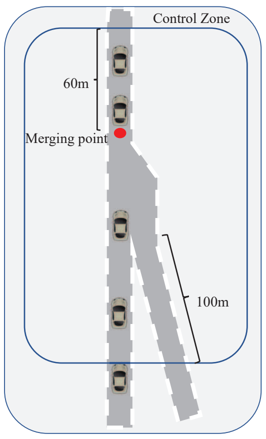

Fig. 3. Diagram of the control zone.

B. State Space

When the vehicle is in the environment, the vehicle needs to observe the state information of the other vehicles in the environment and the state information of the ego vehicle to form the state space. We adopt the description of state space in [12]. The state space as follows:

$$S = (d_{\text{preceding2}}, v_{\text{preceding2}}, d_{\text{preceding1}}, v_{\text{preceding1}}, p_{\text{echo}}, v_{\text{echo}}, a_{\text{echo}}, d_{\text{follwing1}}, v_{\text{follwing1}}, d_{\text{follwing2}}, v_{\text{follwing2}}), \tag{8}$$

Here, $(d_{\text{preceding2}}, v_{\text{preceding2}}, d_{\text{preceding1}}, v_{\text{preceding1}}, d_{\text{follwing1}}, v_{\text{follwing1}}, d_{\text{follwing2}}, v_{\text{follwing2}})$ is the state information of the environment observed by the merging vehicle. $(d_{\text{preceding2}}, d_{\text{preceding1}}, d_{\text{follwing1}}, d_{\text{follwing2}})$ are, respectively, the distance from preceding vehicles of the adjacent lane and following vehicles of the adjacent lane to the ego vehicle. $(v_{\text{preceding2}}, v_{\text{preceding1}}, v_{\text{follwing1}}, v_{\text{follwing2}})$ are, respectively, the velocity of preceding vehicles of the adjacent lane and following vehicles of the adjacent lane. The state information of the merging vehicle $(p_{echo},\ v_{echo},\ a_{echo})$ are, respectively, the position, speed, and acceleration of the ego vehicle, which are measured by sensors. All state information in the state space are normalized to [-1,1].

C. Action Space

In the scenario of ramp merge, the RL algorithm only control the longitudinal direction of the vehicle. The driving trajectory of the vehicle is set by default. The action space is described by the acceleration, and at every simulation step$(0.1s)$. The acceleration is not a random value. Excessive acceleration leads to a decrease in passenger comfort. Therefore, human data on ramp merge are collected on the highway. Fig. 4 shows the acceleration distribution data of human drivers. We choose the range of acceleration $(a \in \left[-3m/s^2, 2.5m/s^2\right])$ on the basis of the experience of passengers.

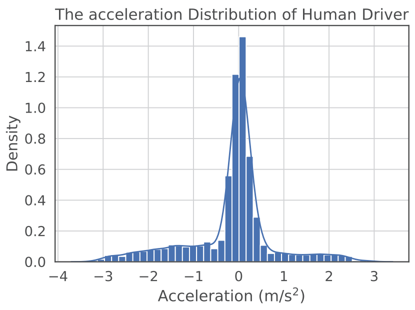

Fig. 4. Acceleration distribution data for human drivers.

D. Reward

After the RL agent performs an action, a single-step reward is obtained. The role of reward is to give the agent a positive signal or a negative signal to obtain the optimal ingestion strategy. The following reward is designed for merging.

1) Speed r_v

In terms of the velocity, the penalty as follows:

$$r_v = \begin{cases} -w_1 * \frac{|v_{ego}-v_{\max}|}{v_{\max}} & \text{if } v_{ego} \leq v_{\max}, \\ -w_2 * (v_{ego} - v_{\max}) & \text{otherwise}, \end{cases} \tag{9}$$

Here, w_1 is the weight to penalize the speed, and w_2 is the weight of the penalty given by w_{max}. w_1 and w_2, respectively, are 0.015 and 0.15. The weights in w_1, w_2 are all debugged during the training process.

2)Termination r_t

Ultimately there are three conditions for the episode to be terminated: 1) the vehicle collides; 2) the vehicle stops; 3) the vehicle successfully merges into the traffic flow. The reward function is described as follows:

$$r_t = \begin{cases} -1 & \text{if collision}, \\ 1 & \text{if success}, \\ -0.4 & \text{if } v_{\text{ego}} = 0. \end{cases} \tag{10}$$

The reward can be described by (9)-(10):

$$r = r_v + r_t. \tag{11}$$

IV. Simulation Experiment

A. Simulation Setup

In this experiment, we use Python to develop scripts and control vehicles in SUMO through the TraCI (Traffic Control Interface). The framework is shown in Fig. 5. The relatively moderate traffic dense is considerd, a vehicle is generated with a probability of 0.5 at every second. The initial state of the vehicles is composed of the random initial speed and random departure lane. The main road vehicles are controlled using the intelligent driver model (IDM) and different parameters to simulate different driver characteristics are set in the real world. According to the parameters, driver characteristics are divided into aggressive, cautious and normal drivers and different driver characteristics are randomly generated in the simulation. At the same time, to simulate the sensor noise in the real traffic environment, noise is injected into the state space, notice that the noise is injected into the detected distance and velocity of the surrounding vehicles, because the state information of the ego vehicle is measured by the own sensors. The ego vehicle is furnished with sensors and detected the information of the surrounding vehicles. Furthermore, the detection range of the lidar is set as 150 m.

B. Results

We compare the RTD3 agent with the TD3 agent in noise-free, mid-level noise (5%), and high-level noise (10%) environments, respectively. Different levels of noise are randomly generated in the state information by a Gaussian distribution. The hyperparameters of the two methods are consistent. The learning rate for the actor network and the critic network, respectively, is 0.0003 and 0.0003, time steps for LSTM = 8. The other hyperparameters are set as follows: replay memory = 1000000, mini batch size = 256, delayed policy update frequency = 2, discount factor = 0.99, soft update coefficient = 0.005, and training episode = 4000 for the ramp merge.

Fig. 6 shows the average reward for both agents in the noise-free environment during the training phase. From the figure, we infer that the RTD3 agent exhibit considerable improvement in the learning speed as compared to the TD3 agent. The TD3 agent converges after 1600 episodes. However,

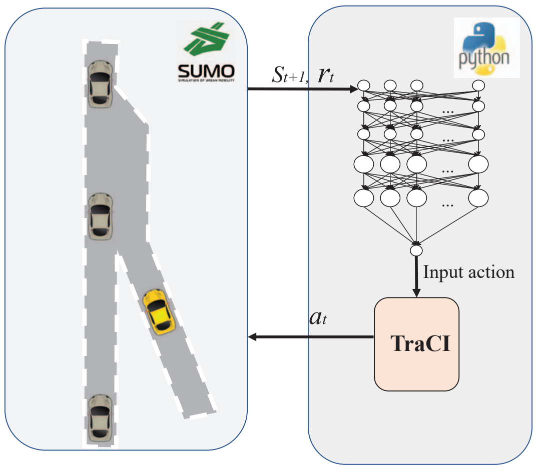

Fig. 5. The simulation framework.

the RTD3 agent converges 600 episodes. The RTD3 agent converges faster than the TD3 agent because it can capture historical information and has long-term memory ability. This reflects that the introduction of temporal information makes the agent converge faster than relying only on the current information.

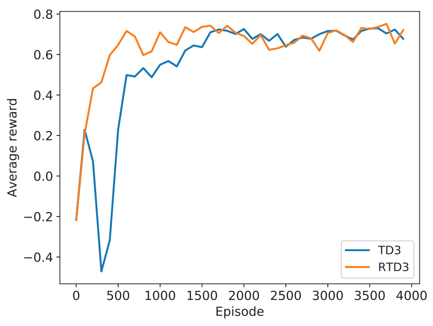

Fig. 6. Training performance of two agents in deterministic environment.

Fig. 7 shows the training time for both agents in the noise-free environment. As can be seen from the figure, the training time of the RTD3 agent and the TD3 agent increases linearly. The RTD3 agent combines with LSTM neural network makes the network structure more complex, which causes the RTD3 agent to spend more time in processing time than the TD3 agent. However, this is completely acceptable relative to the overall training process. Finally, the training time cost of the RTD3 agent is 7080.1s, and the training time cost of the TD3 agent is 5760.7s.

To validate the generalization of two trained agents, two trained agents are evaluated in the noise-free and mid-level noise environment respectively and we evaluate each trained agent for 1000 episodes. Table I shows the performance of two trained agents in the evaluation phase. firstly, two trained agents are compared in a noise-free environment. The results

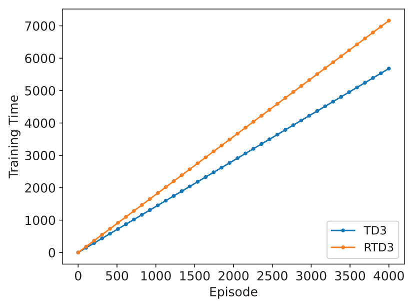

Fig. 7. Training time cost of two agents in deterministic environment.

show that both agents have no collision in the evaluation phase. Also, two trained agents are evaluated in the mid-level noise environment. The results are showed that the RTD3 agent is basically not disturbed by the noisy environment and the collision rate remained within 1%. However, the performance of TD3 agent has a significant degrade in the noise environment. The results show that the collision rate of the TD3 agent dropped 13% compared with the collision rate without noise and the average speed is only 19.2 m/s.

TABLE I
EVALUATE RESULTS WITH NOISE-FREE AND MID-LEVEL NOISE ENVIRONMENT

	Noise-Free		Mid-level Noise (5%)	
	collision free eposides	average speed (m/s)	collision free eposides	average speed (m/s)
TD3	100%	20.8	87%	19.2
RTD3	100%	20.9	99.1%	20.6

To prove the performance of the RTD3 agent in a realistic environment (uncertain sensory data), two agents are trained in the mid-level noise environment. The results are shown in Fig. 8. As we analyzed before, the overall performance of the TD3 agent suffers greatly in noisy scenarios. However, the overall performance of the RTD3 agent does not degrade significantly. This means that the RTD3 agent has good performance in a stochastic environment and is more suitable for a real traffic environment.

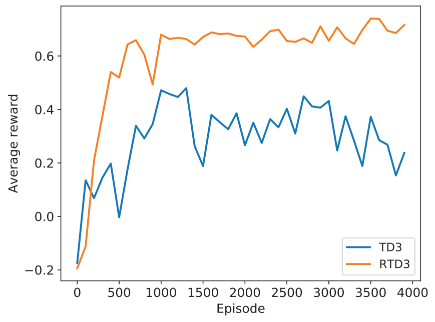

Fig. 8. Training performance of two agents in mid-level noise (5%).

Fig. 9 shows the training time for both agents in the mid-level noise (5%). The training time cost of the RTD3 agent is 7160.5s, and the training time cost of the TD3 agent is 5680.3s. The results indicate that the training time cost of agents is not affected by noise and only by the complexity of the network.

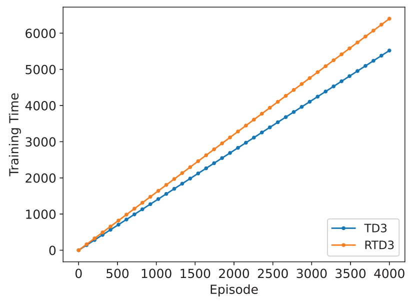

Fig. 9. Training time cost of two agents in mid-level noise (5%).

Similarly, two agents are evaluated in the mid-level noise and high-level noise environment to verify the performance of two agents in different level noise scenarios. Each trained agent is evaluated for 1000 episodes. The results are depicted in Table II. From the table, The results show that the RTD3 agent shows no collision and the average speed is 20.1 m/s in the mid-level noise environment. Also, to evaluate the robustness of the RTD3 agent, the RTD3 agent is evaluated in a high-level noise environment. we observe that the RTD3 agent also has no collisions and the average speed has not dropped significantly in the evaluation phase. However, compared with the RTD3 agent, the TD3 agent learns a bad driving policy and achieves a very high collision rate of 49.9%. This means that the noisy environment has a large influence on the TD3 agent in the training phase, which leads to the fact that even when evaluated in the same level noisy environment, the performance of the TD3 agent trained in a noisy environment is far worse than that in a noise-free environment.

TABLE II
EVALUATE RESULTS WITH MID-LEVEL NOISE ENVIRONMENT AND HIGH-LEVEL NOISE ENVIRONMENT

	Mid-level Noise (5%)		High-level Noise (10%)	
	collision free eposides	average speed (m/s)	collision free eposides	average speed (m/s)
TD3	50.1%	22.2	49.3%	22.1
RTD3	100%	20.1	100%	19.8

The results mentioned above prove that the RTD3 agent has a more robust and reliable performance in noisy environments (closer to real-world situations) and is more suitable for real-world applications than TD3 agent.

V. Conclusion And Discussion

In our work, we design a DRL method to solve the issue of on-ramp merge in uncertain scenarios. The recurrent neural network module is added to the actor network in TD3 to handle temporal sequence. The performance of the proposed method is proved in SUMO. The two methods are compared in different noise environments. The comparison results show that although our method takes more time consumption, it has better ability to deal with uncertain traffic environment and better generalization than the TD3 agent.

Our future work is dedicated to the research on the predict of surround driver's intention and the application of the driver's intention to the reinforcement learning. In addition, we will try out more simulation software, such as the CARLA simulator.

References

[1] G. Li, Y. Yang, S. Li, X. Qu, N. Lyu, and S. E. Li, "Decision making of autonomous vehicles in lane change scenarios: Deep reinforcement learning approaches with risk awareness," *Transportation Research Part C: Emerging Technologies*, vol. 134, p. 103452, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X21004411

[2] S. Chen, J. Hu, Y. Shi, L. Zhao, and W. Li, "A vision of c-v2x: Technologies, field testing, and challenges with chinese development," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 3872–3881, 2020.

[3] F. Pan and H. Bao, "Preceding vehicle following algorithm with human driving characteristics," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 235, no. 7, pp. 1825–1834, 2021.

[4] M. Zhou, X. Qu, and S. Jin, "On the impact of cooperative autonomous vehicles in improving freeway merging: A modified intelligent driver model-based approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 6, pp. 1422–1428, 2017.

[5] M. Shen, H. Hu, B. Sun, and W. Deng, "Heuristics based cooperative planning for highway on-ramp merge," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 1266–1272.

[6] Y. Zhou, M. E. Cholette, A. Bhaskar, and E. Chung, "Optimal vehicle trajectory planning with control constraints and recursive implementation for automated on-ramp merging," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 9, pp. 3409–3420, 2019.

[7] S. Triest, A. Villaflor, and J. M. Dolan, "Learning highway ramp merging via reinforcement learning with temporally-extended actions," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 1595–1600.

[8] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2034–2039.

[9] H. Wang, S. Yuan, M. Guo, X. Li, and W. Lan, "A deep reinforcement learning-based approach for autonomous driving in highway on-ramp merge," *Proceedings of the Institution of Mechanical Engineers Part D Journal of Automobile Engineering*, vol. 235, no. 6, pp. 1–14, 2021.

[10] P. Wang and C.-Y. Chan, "Formulation of deep reinforcement learning architecture toward autonomous driving for on-ramp merge," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 1–6.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[12] Y. Lin, J. McPhee, and N. L. Azad, "Anti-jerk on-ramp merging using deep reinforcement learning," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 7–14.

[13] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1587–1596.

[14] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[15] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wiessner, "Microscopic traffic simulation using sumo," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2575–2582.