Artificial Life and Robotics (2023) 28:158–165
https://doi.org/10.1007/s10015-022-00846-8

ORIGINAL ARTICLE

Autonomous highway driving using reinforcement learning with safety check system based on time-to-collision

Xiaotong Nie[1] · Yupeng Liang[1] · Kazuhiro Ohkura[1]

Received: 9 September 2022 / Accepted: 5 December 2022 / Published online: 26 December 2022
© International Society of Artificial Life and Robotics (ISAROB) 2022

Abstract
Decision making is an essential component of autonomous vehicle technology and received significant attention from academic and industry organizations. One of the promising approaches in designing a decision-making method is Reinforcement Learning (RL). To apply an RL algorithm to an autonomous driving problem, a feature representation of the state must first be chosen. The most commonly used representation is the spatial-temporal state feature. However, if the number or order of the surrounding vehicle changes, the feature representation will be affected. In this paper, we utilize time-to-collision (TTC) as the feature representation and propose a TTC-based safety check system. The action output by the RL controller would be replaced with a safer action chosen by the safety check system when an agent detects a potential collision, i.e., the TTC is below the time threshold. A ramp merging task is used to illustrate the effect. Simulation results show that the proposed method can effectively improve the arrival rate and reduce the collision rate, even in the case of dense traffic situations. Furthermore, we also conducted experiments to examine the performance of the safety check system with different time thresholds.

Keywords Autonomous Vehicle · Reinforcement Learning · Collision Avoidance System

1 Introduction

In recent years, the development of autonomous vehicle (AV) technologies are greatly promoted by advances in the field of artificial intelligence (AI) and machine learning (ML). However, there are still many issues in high interactive traffic scenarios such as ramp merging [1] unprotected left turns [2, 3], and narrow street passing [4, 5]. Autonomous vehicles need to interact with other traffic participants, react to road objects, and select an appropriate strategy.

Most autonomous vehicles have a modular hierarchical structure and can be divided into four components [6], which are perception, prediction, decision-making, and control. Decision-making is an essential component and received significant attention from academic and industry organizations. The majority of current approaches for decision-making methods can be divided into the rule-based method and the data-driven method. The rule-based methods employ heuristics and hard-coded rules to guide the behaviors, such as the Intelligent Driver Model (IDM) [7] and the MOBIL model [8]. For instance, if an autonomous vehicle with a rule-based model observes a stop sign while driving, rules enforce the model to set the acceleration to negative until the vehicle stop. It is feasible to design a strategy hand-crafted for simple traffic scenarios. However, the number of rules increases exponentially in complex scenarios, and there may be conflicts between the rules. Furthermore, the strategies are designed case-to-case, which lacks robustness and generalization ability to new scenarios.

Xiaotong Nie is the presenter of this paper.

This work was submitted and accepted for the Journal Track of the joint symposium of the 28th International Symposium on Artificial Life and Robotics, the 8th International Symposium on BioComplexity, and the 6th International Symposium on Swarm Behavior and Bio-Inspired Robotics (Beppu, Oita, January 25-27, 2023).

✉ Xiaotong Nie
nie@ohk.hiroshima-u.ac.jp

Yupeng Liang
ryo@ohk.hiroshima-u.ac.jp

Kazuhiro Ohkura
kohkura@hiroshima-u.ac.jp

[1] Graduate School of Advanced Science and Engineering, Hiroshima University, Hiroshima, Japan

An alternative is data-driven method such as Reinforcement Learning (RL). Recently, RL has made a remarkable achievement in addressing a variety of robotic problems [9, 10] and autonomous driving tasks [11, 12] . The decision-making problem for autonomous navigation can be formalized as a Markov Decision Process (MDP) [11]. An agent (autonomous vehicle) tries to select the optimal policy to maximize the rewards with consideration of the influences of its behaviors through dynamic interaction with the environment. Most of the previous works are limited to single-agent tasks and cannot be directly introduced to multi-agent tasks. Multi-agent RL algorithms need to maximize the sum of the rewards of all agents, which makes it more difficult to optimize the network. Furthermore, as the number of agents increases, the complexity of the environment rises as well, which leads to a dramatic increase in the variance of optimization methods that estimate gradients by sampling.

To apply a reinforcement learning algorithm to an autonomous driving problem, a feature representation of the state must first be chosen. The state should at least contain a description of nearby vehicles and the environment. The spatial-temporal state feature is the most commonly-used representation [13]. A vehicle driving on the road can be described in a kinematic way by its continuous position, velocity, and heading. This representation is efficient. However, it has two limitations. First, the environment could vary over time and space, which is problematic for learning approaches that expect constant-sized inputs. Second, this type of feature representation is permutation-variant, i.e., dependent on the order in which the interactive agents are listed. For instance, simply switching feature entries of agent i and agent j would result in a different feature representation.

In this paper, we utilize Time-to-Collision (TTC) as the feature representation to train a controller for multiple autonomous vehicles. TTC is the time needed for a vehicle to collide if it continues driving on the same route and at the same speed. TTC focuses on the potential risk posed by other vehicles and static obstacles rather than a specific agent. Therefore, even if the number or order of the surrounding vehicles changes, the feature representation will not be affected.

We also propose a safety check system (SCS) based on TTC as shown in Fig. 1. First, the SCS needs to determine whether a vehicle is unsafe. TTC is used as a threat assessment in several approaches [14, 15]. However, the general definition of TTC is calculated from relative distance and relative velocity with constant relative acceleration. When two vehicles are moving with approximately the same velocity, even if the distance between them is very close, the general TTC-based SCS will not detect a potential collision. Therefore, in the proposed method, the TTC under the three driving circumstances of uniform speed, acceleration, and deceleration will be calculated to improve the safety of the system. Second, the SCS needs to replace dangerous action output by RL with a safer action. For instance, the safety system proposed in [16, 17] includes a dynamically learned safety module in addition to handcrafted safety rules. However, the majority of current safety enhancement methods were created for single autonomous vehicle tasks. For tasks involving multiple autonomous vehicles, the purpose of SCS is to guarantee the safety of the entire system. In this case, relying on communication between vehicles, the proposed method can satisfy comprehensive requirements including safety and order rationality.

The main contributions of this paper are summarized as follows.

1. We use the modified TTC as the state representation to train an RL controller for multiple AVs. It performs better than the conventional approach utilizing kinematics, demonstrating the reliability of this state representation.

2. We propose a safety check system for multiple AVs to enhance the safety of the system and improve the learning efficiency of RL. Simulation results show that the proposed method can effectively improve the arrival rate and reduce the collision rate, even in the case of dense traffic situations. Furthermore, we also conducted experiments to examine the performance of the safety check system with different time thresholds.

3. A ramp merging task in the computer simulation is used to examine the effects of the proposed method. Autonomous vehicles and environmental vehicles co-exist in the merge lane and the main lane. Vehicles on the ramp need to merge into the main lane efficiently without collision. After passing the main lane, autonomous vehicles need to divert to the ramp and reach the goal.

The remainder of this paper is organized as follows. Section 2 explains the methods. Experiment settings are described in Section 3. Results are shown in Sect. 4. Finally, we conclude this paper in Sect. 5.

2 Methods

In this study, a deep reinforcement learning (DRL) algorithm called Proximal Policy Optimization (PPO) [18], is used to train a decision-making controller. However,

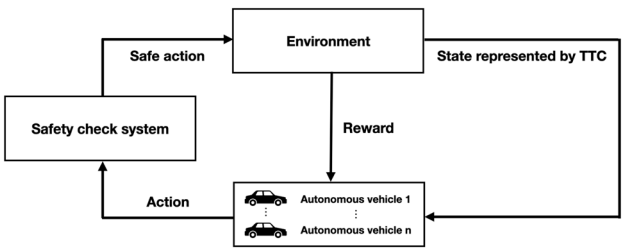


Fig. 1 The framework of the proposed approach

DRL algorithms are not safe enough since the agent is encouraged to explore a wide range of states to find the best strategy. Therefore, it is necessary to equip autonomous vehicles with a security assurance mechanism when collisions are about to occur. In this study, we propose a safety check system for reducing collisions by utilizing Time-To-Collision (TTC).

2.1 Proximal policy optimization (PPO)

A reinforcement learning problem can be formulated as a Markov Decision Process (MDP), which involves a set of states S, a set of actions A, a reward function R: $S \times A \rightarrow \mathbb{R}$, and a probability function of the state transition P: $S \times S \times A \rightarrow [0,1]$. At each timestep, the agent observes a state $s \in S$, takes an action $a \in A$, and gets a reward $r \in \mathbb{R}$, then moves to the state $s' \in S$. The final objective is to find an optimal control policy $\pi : S \rightarrow A$, which maximizes the long-term cumulative reward. One of the most popular reinforcement learning algorithms, Proximal Policy Optimization (PPO) [18], is applied in this task. It achieves remarkable levels of performance in a wide range of tasks including Atari 2600 games and Mojoco continuous control tasks. This method has two networks, i.e., actor and critic networks. PPO trains a stochastic policy in an on-policy way. It improves the efficiency of the training data with the implementation of the clip function in the objective function. The main contribution of PPO is to make sure that an updated policy does not depart too far from the previous policy. This leads to smooth training and ensures that the agent will improve the performance monotonously.

The objective function for PPO is shown in Eq. 1.

$$L^{\mathrm{CLIP}}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \quad (1)$$

Where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ denotes the probability ratio under new and old policies, ϵ is a hyperparameter denotes the limit of the range within which the update is allowed, $\hat{A}_t$ is the advantage of current timestep.

PPO ensures that in each update, the new policy is not too different from the old policy. It acheives this by clipping the probability ratio r_t in the range of $[1-\epsilon, 1+\epsilon]$. In this way, a huge update that might potentially be irrecoverably harmful is not allowed.

2.2 Time-to-collision (TTC)

In research on traffic conflicts techniques, Time-To-Collision (TTC) has proven to be an effective measure for rating the severity of traffic conflicts [19]. According to Hayward's definition, TTC [20] is the amount of time needed for two vehicles to collide if they continue driving in the same route and at the same speed. It stands for the danger posed by the vehicle at the current lane and speed. There is a higher chance of a collision when the TTC is low. The TTC of two vehicles can be approximated by Eq. 2.

$$TTC = \frac{|R_i|}{|V_i| \cdot \cos < R_i, V_i >} \quad (2)$$

Where R_i is the relative position vector of vehicle i, V_i is the relative velocity vector of vehicle i, $|\cdot|$ is the 2-norm of a vector.

2.3 PPO with safety check

In this study, we propose a safety check system based on TTC. The central concept is that the action output by the DRL controller should be replaced with a safer action chosen by the safety check system when an autonomous vehicle detects a potential collision, i.e., the TTC is below the threshold. The overall algorithm is shown in Algorithm 1.

Algorithm 1 PPO with safety check

```
Inistialize replay buffer D.
for m = 0 to M episodes do
    for t = 0 to T timesteps do
        for v = 0 to V vehicles do
            Observe s_t^v.
            Select an action a_t^v using the neural network.
        end for
        Safety Check
        for v = 0 to V vehicles do
            Execute new action a_t^v' in environment.
            Get reward r_i^v.
            Add to replay buffer D with (s_t, a_t^v', r_i^v)
        end for
    end for
    Update the networks using PPO.
end for
```

The detection of collision is achieved by calculating the TTC under the three driving circumstances of uniform speed, acceleration, and deceleration. Autonomous vehicles will communicate with each other, when a potential collision is detected, a safer action using Algorithm 2 will replace the action output by the RL controller. First, all the AVs will calculate their TTC according to the action output by the RL controller. Then, each AV broadcasts its TTC to other AVs and sequences them based on the level of risk. The action will be replaced if the TTC is below the time threshold. The high-risk vehicle will be given top priority for action replacement. The TTC of every possible action will be recalculated and the action with the highest TTC will be chosen as the new action, indicating maximum safety. When the

new action is selected, the high-priority AV will broadcast its latest target lane and speed to others. The process will then be repeated by vehicles with lower priority until the entire system is in a safe situation.

Algorithm 2 Safety check

for each autonomous vehicle v **do**
  Calculate the TTC if excuate action a_t^v .
  Broadcast its TTC to all neighboring autonomous vehicles.
  v.decided = False.
end for
Within each autonomous vehicle, sequence the vehicles by TTC from small to large.
for each autonomous vehicle v with lowest TTC **do**
  v.decided = True.
  Calculate TTCs with different actions using the target lane and speed of other vehicles whose decided is True, and ignoring the autonomous vehicles whose decided is False.
  if TTC of the current speed $<$ Threshold **then**
    Select the action with highest TTC.
  end if
  Calculate the target lane and speed with the new action.
  Send the target lane and speed to all other autonomous vehicles.
end for

3 Experiment

This study conducted experiments in highway-env simulators [21]. The code is implemented in Python using Pytorch framework. A desktop computer with NVIDIA RTX 3070 GPU, AMD Ryzen 9 3950x CPU, and 128GB memory is used for the experiments.

3.1 Task settings

In this study, a ramp merging task is used to evaluate our method. The environment is shown in Fig. 2. The simple task consists of 4 autonomous vehicles and 6 environmental vehicles. There are 300,000 timesteps in one training process. The hard task consists of 8 autonomous vehicles and 8 environmental vehicles. There are 500,000 timesteps in one training process. To increase the complexity of the hard task, an obstacle is added to the main lane. In the beginning, the vehicles in each of the three lanes are generated at a random position. Autonomous vehicles and environmental vehicles co-exist in the merge lane and the main lane. Vehicles on the ramp need to merge into the main lane efficiently without collision. After passing the main lane, autonomous vehicles need to divert to the ramp and reach the goal. For environmental vehicles, we utilize the Intelligent Driver Model (IDM) [7] and MOBIL model [8] for longitudinal acceleration and lateral lane change decisions.

3.2 Neural network settings

The state space of the baseline method is the features of other vehicles, including *ispresent*, x, y, v_x, v_y, where *ispresent* is a variable denoting whether a vehicle is observable, x and y are the longitudinal and lateral position of the observed vehicle, v_x and v_y are the longitudinal and lateral speed of the observed vehicle. A maximum of 16 vehicles (including 8 autonomous vehicles and 8 environmental vehicles) can be observed, the state is represented by a 16×5 matrix.

The proposed approach uses TTC as the state representation. At each timestep, each vehicle will calculate TTCs with different speeds. In this study, the state of the autonomous vehicle is represented by a $3 \times 3 \times 10$ matrix. The first dimension represents the TTC if the vehicle at the current speed, slows down, or speeds up. The second dimension represents the left, current, and right lane of the vehicle.

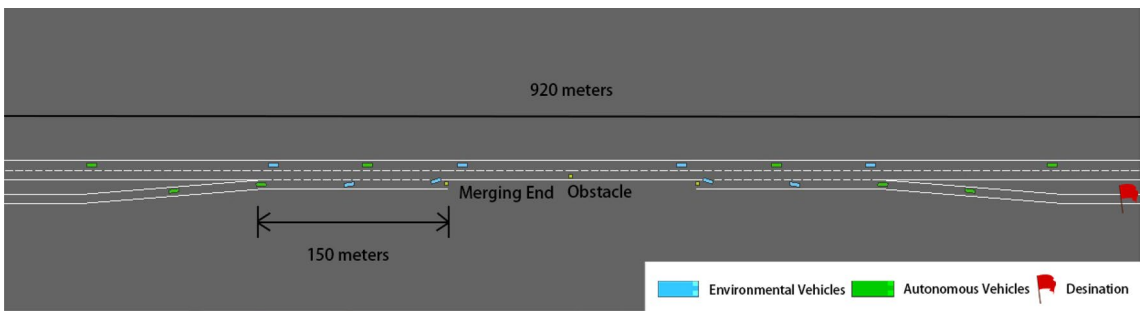


Fig. 2 The ramp merging task conducted using the highway-env simulator. In the left lanes, the vehicles are generated at random. Environmental vehicles are depicted in blue, whereas autonomous vehicles are depicted in green. The on-ramp's vehicles need to merge into the main lane. After a section of main lane, autonomous vehicles need to exit the highway by the off-ramp

The third dimension represents the TTC time in one-hot encoding.

The action space of the autonomous vehicle is defined as the set of high-level control decisions, including turn left, idle, turn right, speed up, and slow down. After selecting the high-level decision, the low-level controller will produce the corresponding steering and throttle control signals to control the vehicle.

There are two neural networks in PPO, i.e., the actor and the critic. Fully connected layers are used in both neural networks. The inputs of the actor and the critic networks are the states of the autonomous vehicle. The actor's outputs are the probabilities of the actions that the vehicle can take. The output of the critic network is the value function of the current state. There are two fully connected layers in the actor and critic networks, each of which has a hidden unit of 256.

The hyper-parameters used in this study are shown in Table 1. Each experiment was repeated ten times with different random seeds.

3.3 Reward settings

The performance of deep reinforcement learning algorithms is highly dependent on the design of the reward functions. At each timestep, each vehicle gets a reward using the Equation 3.

$$r_t = r_c + r_a + r_s + r_l \tag{3}$$

Where r_t is the total reward that the vehicle can receive at each timestep, r_c is the collision penalty when the vehicle is involved in a collision, r_a is the arrived reward when the vehicle reaches the goal, r_s is the reward when the vehicle speeds up. The penalty for changing lanes is r_l, which is intended to discourage lane switching by the vehicle. The values of each reward setting are shown in Table 2.

Table 1 Hyper-parameters

Hyper-parameter	Value
Batch size	64
Buffer size	240
Parallel environments	32
Optimizer	Adam
Learning rate	0.0005
Number of timesteps	3e5
ϵ in PPO	0.2
TTC time threshold	3
Input size of kinematic representation	16×5
Input size of TTC representation	$3 \times 3 \times 10$
Output size	5
Hidden layers size of actor network	[256,256]
Hidden layers size of critic network	[256,256]

4 Results

4.1 Simple task

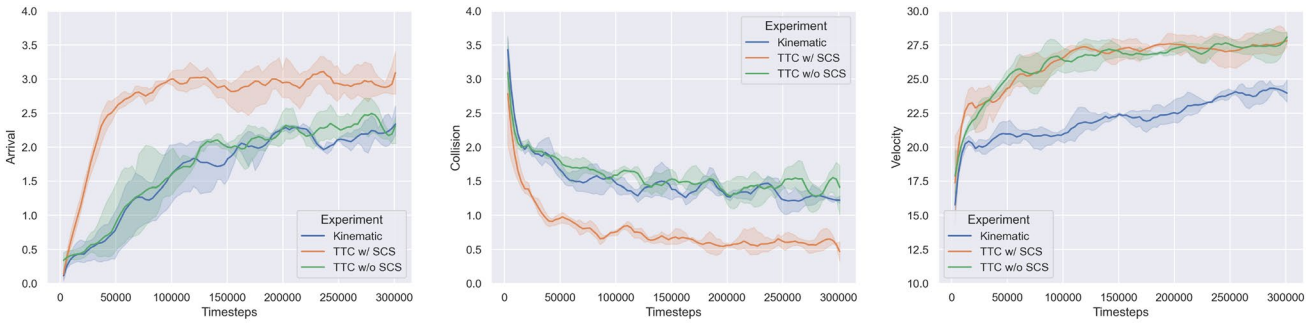
Ablation experiments are conducted to investigate the impact of the state representation and the safety check system. We employ the conventional PPO method as the baseline method, which does not employ the safety check system and uses kinematic representation. It needs to be noted that the conventional method cannot equip the safety check system due to a lack of TTC information. Figure 3 shows the performance trajectories on the simple task, which is an average of 10 trials. The simple task consists of 4 autonomous vehicles and 6 environmental vehicles. Figure 3a shows the number of arrivals for the conventional method, the method using TTC representation without the safety check system, and the proposed method. Arrivals converge to 2.31, 2.33, and 3.1 respectively. When the safety check system is not provided, the conventional method and the method using the TTC representation have similar performance. Figure 3b shows the number of collisions for the conventional method, the method using TTC representation without the safety check system, and the proposed method. Collisions converge to 1.22, 1.41, and 0.47 respectively. The best collision avoidance ability can be obtained by the proposed method. Figure 3c shows the velocity for the conventional method, the method using TTC representation without the safety check system, and the proposed method. The velocity converges to 23.9m/s, 28.1m/s, and 27.8m/s respectively. Simulation results show that the proposed method can effectively improve the arrival rate and reduce the collision rate without reducing the efficiency of autonomous vehicles.

4.2 Hard task

Figure 4 shows the performance trajectories on the hard task, which is an average of 10 trials. The hard task consists of 8 autonomous vehicles and 8 environmental vehicles. An obstacle is in the middle of the main lane. Figure 4a shows

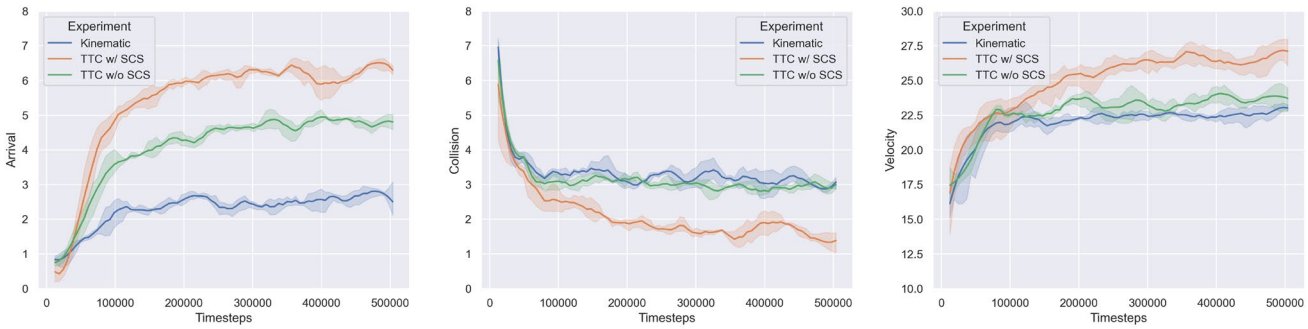
Table 2 Reward settings

Reward	Value
Collision Reward r_c	−50
Arrived Reward r_a	100
High Speed Reward r_s	0.2
Lane Change Reward r_l	−0.05



(a) The number of arrivals. **(b)** The number of collisions. **(c)** Velocity of autonomous vehicles.

Fig. 3 The performance trajectories in the simple task, where each data point is the average of 10 trials (with standard deviation)



(a) The number of arrivals. **(b)** The number of collisions. **(c)** Velocity of autonomous vehicles.

Fig. 4 The performance trajectories in the hard task, where each data point is the average of 10 trials (with standard deviation)

the number of arrivals for the conventional method with kinematic representation, the method using TTC representation without the safety check system, and the proposed method. Arrivals converge to 2.5, 4.8, and 6.3 respectively. As mentioned earlier, kinematic state representation would be affected by the number or order of surrounding vehicles. The effect would be magnified in a dense traffic situation. As a result, the conventional method performs worse than methods using TTC representation, even without the safety check system. Figure 4b shows the number of collisions for the conventional method, the method using TTC representation without the safety check system, and the proposed method. Collisions converge to 3.0, 3.0, and 1.4 respectively. The proposed method has the best collision avoidance ability. Figure 4c shows the velocity for the conventional method, the method using TTC representation without the safety check system, and the proposed method. The velocity converges to 23.7m/s, 23.0m/s, and 27.1m/s respectively. In a dense traffic environment, the speed of each algorithm decreases slightly more than in the simple task. Generally, simulation results show that the proposed method can deal with a dense traffic scenario more than other approaches.

Simulation snapshots of the controller trained with the proposed method are shown in Fig. 5. It can be observed in Fig. 5a, b, that the first autonomous vehicle is driving on the main road. The second autonomous vehicle is driving on the merging road. As shown in Fig. 5c, when the distance between the two vehicles is close, which means the TTC is less than the safe time threshold, the action output by DRL will be replaced by a safer action. Therefore, the first autonomous vehicle chooses to turn left into another lane to avoid the potential collision, rather than go straight to maximize the expected return. In Fig. 5d, e, the second vehicle successfully merges into the main lane. As seen in Fig. 5f, the second vehicle makes a lane change to avoid a probable collision with an environmental vehicle.

Furthermore, to examine the effect of different safety time thresholds, evaluation experiments were performed. The safety time threshold is an indicator for evaluating the degree of danger of a state. When TTC is less than the safety time threshold, the actions output by the RL method will be replaced with safer actions by the safety check system. Therefore, we changed the safety time thresholds to 1s, 3s, 5s, 7s, and 9s, respectively. The corresponding controllers are trained in the

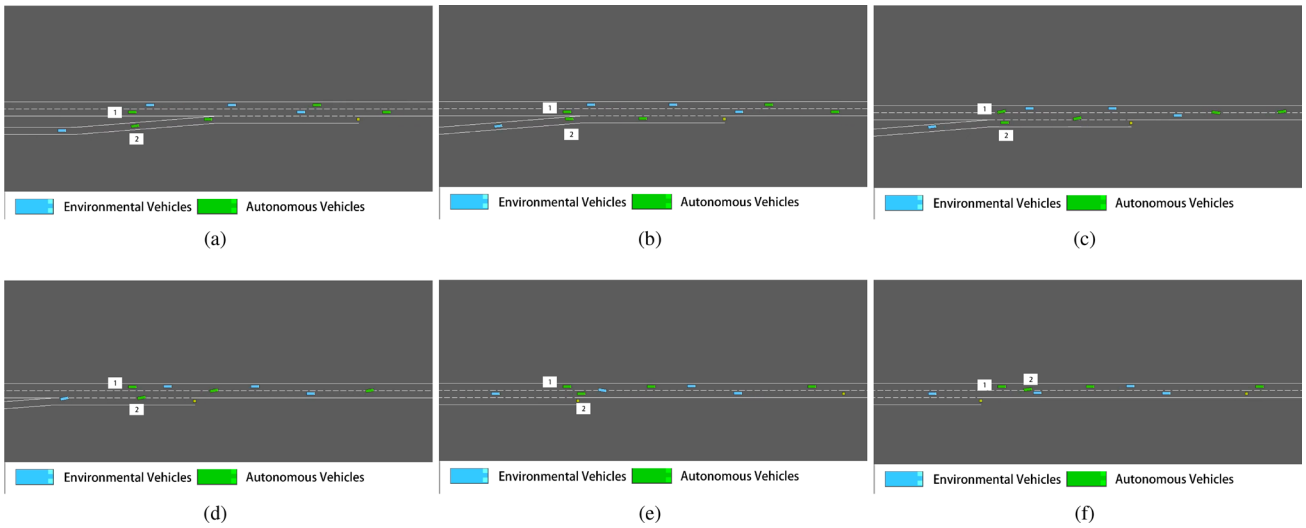


Fig. 5 Simulation snapshots of the controller trained by the proposed method

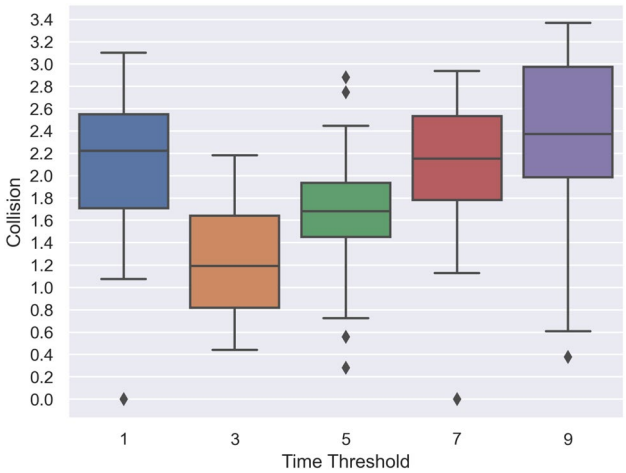


Fig. 6 Collision evaluation experiments with different safety time thresholds

hard task. Each controller is evaluated 50 times. The results of the evaluation experiments are shown in Fig. 6. As it can be observed, when the time threshold is 3s, the lowest collision can be obtained. As the safety time threshold increases, the number of collisions increases accordingly. Since it is still relatively safe when TTC is 7s or 9s, it is not ideal to replace actions too early. Especially there is a great fluctuation when the time threshold is 1s. It is difficult to completely avoid a collision since the emergency response time is insufficient. As a result, the best performance and safety can be obtained by setting the safety time threshold to 3s.

5 Conclusion

In this paper, we propose a safety check system based on time-to-collision. The central concept is that the action output by the DRL controller should be replaced with a safer action chosen by the safety check system when an agent detects a potential collision, i.e., the TTC is below the time threshold. A ramp merging task is used to examine the effects of the proposed method.

Simulation results show that the proposed method can effectively improve the arrival rate and reduce the collision rate, even in the case of dense traffic situations. Furthermore, we also conducted experiments to examine the performance of the safety check system with different time thresholds.

One limitation of our approach is the lack of complexity in the design of the simulated environment. In the future scope, more research on autonomous vehicles will be conducted. For instance, developing a more realistic simulation environment by incorporating data from real-world traffic systems could be an interesting direction.

Acknowledgements This work was supported by Initiative for Realizing Diversity in the Research Environment (Specific Correspondence Type).

References

1. Riccardo S, Benjamin H (2014) Control concepts for facilitating motorway on-ramp merging using intelligent vehicles. Trans Rev 34(6):775–797

2. Wang X, Zhao D, Peng H, LeBlanc DJ (2017) Analysis of unprotected intersection left-turn conflicts based on naturalistic driving data. In: 2017 IEEE Intelligent Vehicles Symposium (IV), pages 218–223. IEEE
3. Xuedong Y, Essam R (2007) Effect of restricted sight distances on driver behaviors during unprotected left-turn phase at signalized intersections. Transp Res Part F: Traffic Psychol Behav 10(4):330–344
4. Saad Y, Zaid N, Norgate Sarah H (2017) Narrow lanes and their effect on drivers' behaviour at motorway roadworks. Transp Res Part F: Traffic Psychol Behav 47:86–100
5. Alexandra K, Hale David K, Mohamadamin A, Bastian S, Anxi J, Joe B (2019) Evaluating the operational effect of narrow lanes and shoulders for the highway capacity manual. Transp Res Record 2673(10):558–570
6. Paden B, Cáp M, Yong SZ, Yershov DS, Frazzoli E (2016) A survey of motion planning and control techniques for self-driving urban vehicles. CoRR. arXiv:abs/1604.07446
7. Martin T, Ansgar H, Dirk H (2000) Congested traffic states in empirical observations and microscopic simulations. Phys Rev E 62(2):1805–1824
8. Arne K, Martin T, Helbing D (1999) General lane-changing model mobil for car-following models. Transport Res Record 1:86–94
9. Gu S, Holly E, Lillicrap T, Levine S (2017) Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In: 2017 IEEE international conference on robotics and automation (ICRA)
10. Jens K, Andrew BJ, Jan P (2013) Reinforcement learning in robotics: a survey. Int J Robot Res 32(11):1238–1274
11. Li Haoran, Zhang Q, Zhao D (2020) Deep reinforcement learning based automatic exploration for navigation in unknown environment. CoRR. arXiv:abs/2007.11808
12. Leurent E, Mercat J (2019) Social attention for autonomous decision-making in dense traffic. CoRR. arXiv:abs/1911.12250
13. Wang W, Wang L, Zhang C, Liu C, Sun L (2022) Social interactions for autonomous driving: a review and perspective
14. Jansson J (2005) Collision avoidance theory: with application to automotive collision mitigation. PhD thesis. Linköping University Electronic Press
15. Noh S, Han W-Y (2014) Collision avoidance in on-road environment for autonomous driving. In: 2014 14th International Conference on Control, Automation and Systems (ICCAS 2014), pages 884–889. IEEE
16. Baheri A, Nageshrao S, Tseng HE, Kolmanovsky I, Girard A, Filev D (2020) Deep reinforcement learning with enhanced safety for autonomous highway driving. In: 2020 IEEE Intelligent Vehicles Symposium (IV), pages 1550–1555
17. Nageshrao S, Tseng HE, Filev D (2019) Autonomous highway driving using deep reinforcement learning. In: 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), pages 2326–2331
18. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal policy optimization algorithms. arXiv:1707.06347
19. Van Der Horst R, Hogema J (1993) Time-to-collision and collision avoidance systems
20. Hayward JC (1972) Near miss determination through use of a scale of danger
21. Leurent E (2018) An environment for autonomous driving decision-making. https://github.com/eleurent/highway-env

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.