Research Article

Multipolicy Robot-Following Model Based on Reinforcement Learning

Ning Yu ^{1,2,3,4} Lin Nan ^{1,2,3} and Tao Ku ^{1,2,3}

¹Key Laboratory of Networked Control Systems, Chinese Academy of Sciences, Shenyang 110016, China

²Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China

³Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang 110169, China

⁴University of Chinese Academy of Sciences, Beijing 100049, China

Correspondence should be addressed to Tao Ku; kutao@sia.cn

Received 11 October 2021; Accepted 19 October 2021; Published 8 November 2021

Academic Editor: Tongguang Ni

Copyright © 2021 Ning Yu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

We propose in this paper a new approach to solve the decision problem of robot-following. Different from the existing single policy model, we propose a multipolicy model, which can change the following policy in time according to the scene. The value of this paper is to obtain a multipolicy robot-following model by the self-learning method, which is used to improve the safety, efficiency, and stability of robot-following in the complex environments. Empirical investigation on a number of datasets reveals that overall, the proposed approach tends to have superior out-of-sample performance when compared to alternative robot-following decision methods. The performance of the model has been improved by about 2 times in situations where there are few obstacles and about 6 times in situations where there are lots of obstacles.

1. Introduction

Robot [1] is a topic of general interest in the automotive industry. It is a key problem that the agent attains a suitable decision of following. At present, there are two methods to solve this question, including the algorithm based on rule and the algorithm based on machine learning [2].

The algorithm based on rule, in general, makes a decision by establishing the appropriate kinematics model, so that the agent shows perfect performance in the predictable environment but may be not adjusted to the new environment. Gao et al. [3] proposed a kind of vehicle cruise system based on simulating driver method, which can safely follow the vehicle on most occasions but is hard to ensure safety and stability at the same time when it is used in complex scenes because it is governed by the driving experience of the designer. Li et al. [4] proposed a feedback control method, which has excellent following efficiency and safety performance. However, it lacks the consideration of stability in the situation of frequent speed changes.

One of the algorithms based on machine learning is based on deep learning [5] and the other is based on reinforcement learning [6]. Wang et al. [7] proposed a robot-following system based on deep neural network [8]. Compared with the traditional following model, it can adapt to complex situations better. This method, however, works by fitting the existing data, rather than generating an optimized strategy. So, it will be limited by the size of the training data set, and it sometimes fails to achieve the optimal decision in the new environment.

The algorithm based on reinforcement learning is that the agent explores continuously the environment and obtains the optimal decision by trial and error [9]. Gao et al. [10] proposed a following system based on Q-learning, which does not need a specific training set and has the ability of continuous learning. However, this method cannot respond immediately while the leading car changes its speed rapidly due to the discreteness of the algorithm. Deng et al. [11] proposed an improved DDPG algorithm. It transforms a multiobjective problem into a single objective problem by fixed weighting found by the

experiment method. This method has better stability performance compared with the above methods and faster training speed compared with the traditional DDPG algorithm. However, this method finds out the real optimal weight only through a large number of simulation experiments, which is difficult to achieve in real scenes. Gao et al. gave a complete follow-up system [12] and proposed a method based on inverse reinforcement learning [13], which solves the problem that the reward function is difficult to set in the following process. Meanwhile, they conducted experiments on real scenes and verified the effectiveness of the system. Even so, he did not consider the diverse needs of robot-following decisions in different environments.

At present, the following problems still exist in the robot-following decision-making problem:

- (1) It is difficult to obtain the optimal decision in different environments by the single decision-making method of robot-following
- (2) It is difficult to properly weigh the relationship between various targets
- (3) Most methods lack consideration of flexible targets such as robot-following stability

In order to solve the above problems, this paper proposes the multipolicy model based on reinforcement learning. This method solves the following problems such as setting reward function hardly, various users' needs, and unstable operation. It achieves the effect of surpassing the current many advanced methods in the simulation environment, basically achieves the target of safe, efficient, and stable following, and can quickly adjust the policy to deal with different environments. Empirical investigation on a number of datasets reveals that overall, compared to alternative robot-following decision methods, this method can reduce the safety risk by 66% and improve the stability performance by 6 times without affecting the robot-following efficiency when the speed of the target changes frequently. Meanwhile, the safety risk can be completely avoided, and the stability performance can be improved by 2 times when the target moves smoothly.

2. Model and Algorithm

2.1. Markova Model. Markova model is a model that the probability of the next action, which is usually used in reinforcement learning modeling, is only related to the current state, but not to the previous action and state. This model can be represented by a five-tuple (S, A, P, R, γ) [14]. S is the set of current state. A is the set of action at the next moment. P is the probability from S to A . R is the reward function. γ , ranging from 0 and 1, is the discount factor, which is used to calculate the cumulative reward value. Policy Π is the probability that the state in set S maps to every action in set $A: S \rightarrow P(A)$. Provided that policy is $\pi_\theta(a, s)$, the cumulative reward value is $J(\theta) = E_{s \sim p^\pi, a \sim \pi_\theta} [\sum_{t=0}^{\infty} \gamma^t r^t]$. Reinforcement learning attains the optimal policy by maximizing.

Firstly, the robot-following process is discretized, and the sampling interval is τ . When time is recorded as t , the distance between the end-effector and the target is recorded as d_t , the relative velocity is recorded as Δv_t , the acceleration of the end-effector is recorded as a_t^2 , and the state of t is recorded as $s_t, s_t(d_t, t, \Delta v_t, a_t^2) \in S$.

After getting the information of the state s_t , the end-effector outputs the action μ_t and attains the reward $R_t = f(\mu_t)$. Meanwhile, the state transforms s_t into s_{t+1} . μ_t is determined by policy Π .

2.2. Algorithm. The algorithm is a reinforcement algorithm based on the Actor-Critic structure [15]. The true Q function [16] is replaced by $Q^{\theta^Q}: S \times A \rightarrow R$, and the target function is as [17]:

$$J(\theta^Q) = \max_{\theta^Q} E_{\beta} \left[\frac{1}{2} \left(r^t + \gamma Q^{\theta^Q}(s_{t+1}, a_{t+1}) - Q^{\theta^Q}(s_t, a_t) \right)^2 \right], \quad (1)$$

$$J(\theta^\mu) = \max_{\theta^\mu} E_{\beta} \left[Q^{\theta^Q}(s_t, \mu(s_t)) \right]. \quad (2)$$

The pseudocode of the algorithm is shown in Algorithm 1.

2.3. Pilot Experience Structure. Pilot experience structure is an important structure to reflect users' preferences and is also an important factor that affects the reward function.

The pilot experience structure consists of experience network and auxiliary network. Experience network is determined by the driving experience of the designer and auxiliary network is determined by users' preferences. The initial policy will be determined by the experience network and auxiliary network. The agent will generate multiple policies by adjusting the influence of the auxiliary network.

2.4. Reward Function. Reward function plays an important role in the reinforcement learning algorithm. A good reward function can accelerate the convergence of the algorithm and improve the performance of the policy. However, it may be hard to find out a good reward function when people face some problems, especially multiobjective optimization problem, in that how to balance the relationship between various targets is difficult to solve. This paper proposes a self-learning method of achieving reward function. The following will introduce the reward function of each single target and the structure of the self-learning method and multipolicy model.

2.4.1. Safety. Safety is the primary target. In general, keeping a reasonable safety distance can ensure safety performance, so the reward function of safety is defined as

$$R_1 = \begin{cases} 0, & d \geq D, \\ -1, & d \leq D, \end{cases} \quad (3)$$

```

Randomly initialize critic network  $Q(s, a|\theta^Q)$  and actor  $\mu(s|\theta^\mu)$  with weights  $\theta^Q$  and  $\theta^\mu$ 
Initialize target network  $Q'$  and  $\mu'$  with weights  $\theta^{Q'} \leftarrow \theta^Q$ ,  $\theta^{\mu'} \leftarrow \theta^\mu$ 
Initialize replay buffer  $R$ 
for episode = 1,  $M$  do
    Initialize a random process  $\mathcal{N}$  for action exploration
    Receive initial observation state  $s_1$ 
    for  $t = 1, T$  do
        Select action  $a_t = \mu(s|\theta^\mu) + N_t$  according to the current policy and exploration noise
        Execute action  $a_t$  and observe reward  $r_t$  and observe new state  $s_{t+1}$ 
        Store transition  $(s_t, a_t, r_t, s_{t+1})$  in  $R$ 
        Sample a random minibatch of  $N$  transitions  $(s_i, a_i, r_i, s_{i+1})$  from  $R$ 
        Set  $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'}))|\theta^{Q'}$ 
        Update critic by minimizing the loss:  $L = (1/N) \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$ 
        Update the actor policy using the sampled gradient:
         $\nabla_{\theta^\mu} \mu|s_i \approx (1/N) \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|s_i$ 
        Update the target networks:
         $\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}$ 
         $\theta^{\mu'} \leftarrow \tau \theta^\mu + (1 - \tau) \theta^{\mu'}$ 
    end for
end for

```

ALGORITHM 1: The pseudocode of the algorithm.

where D is the safety distance.

2.4.2. Stability. Stability generally means that the robot should keep running smoothly at any time, and the speed should not change frequently. Therefore, the stability can be described by acceleration and acceleration change rate, so the reward function of stability is defined as

$$R_2 = \frac{1}{1 + |a_t|} + \frac{1}{1 + |\Delta a / \Delta t|}, \quad (4)$$

where a_t is the acceleration and $\Delta a_t / \Delta t$ is the acceleration change rate.

2.4.3. Efficiency. Efficiency mainly refers to the good cooperation between the robot and the target at any time. At the same time, if possible, the robot should maintain a high speed to ensure less time to complete the task.

Therefore, the reward function for efficiency can be defined as

$$R_3 = \begin{cases} \frac{v_t}{d(1 + \Delta d)}, & d \geq D, \\ 0, & d < D, \end{cases} \quad (5)$$

where v_t is the current velocity, d is the relative distance, Δd is the change of the relative distance, and D is the safety distance.

2.4.4. Self-Learning Method and Multipolicy Model. There is clearly a conflict between these targets. When the relative distance is small, the efficiency is high but there is a risk that the distance is less than the safety distance. When the speed changes small, the stability is high but the relative distance

changes greatly, so the efficiency and safety are difficult to guarantee. How to determine the weight relationship between the targets is the key to integrating into a reward function. Thus, we propose the self-learning method and multipolicy model.

Firstly, several policies are given by the pilot experience structure. The agent will first use these policies for training, and at the same time, it will sample the state-action transition. After many times of training in the training environment, the user needs to give scores of every single target and a comprehensive score. The agent will convert the scores into the total reward of every training and then the reward function can be estimated according to the total reward and the distribution of state-action transition. The maximum likelihood estimation method can be used to estimate the parameter of the reward function. Likelihood function can be defined as

$$L(\mu) = \prod_{i=1}^n p(x_i; \mu). \quad (6)$$

Then, we can take logarithms on both sides of the likelihood function as

$$\begin{aligned} \ln L(\mu) &= \ln \prod_{i=1}^n p(x_i; \mu) \\ &= \sum_{i=1}^n \ln p(x_i; \mu). \end{aligned} \quad (7)$$

Then, we can derive the function $\ln L(\mu)$ with respect to the variable μ and make it equal to 0 as

$$\frac{d \ln L(\mu)}{d\mu} = 0. \quad (8)$$

Then, the estimated value of the parameter can be solved.

By the estimated value of the parameter μ , the probability distribution of the optimal action can be obtained. Then, the

reward function can be derived from the optimal action sequence.

The agent will train a new policy from the new reward function. Similarly, the user needs to score the new policy again. If the comprehensive score is lower than before, the new policy will be added to the training data to estimate a better policy until the score of the new policy exceeds the scores of any policy before.

Since the policy function that meets the requirements is not unique, we can obtain several different value functions. In order to meet the various needs of different users for every target in the multiobjective problem, the agent will train several reward functions and policies which surpass the policies given by the pilot experience structure through the above method. Meanwhile, the agent will classify the policies according to the reward value of the subobjectives which can be calculated by the subevaluation function, which help itself choose the suitable policy in time. The subevaluation function can be attained by the subobjective reward function.

3. Experiment

To verify the validity of the model, two sets of trained policies (stability/efficiency) according to some users' needs are compared with other existing models in three different robot-following simulation scenarios.

In the following experiments, because the maximum measure distance of the distance sensor is 100 cm, we set $d_{\max}$ to 100 cm. Similarly, $v_{\max}$ is set to 55 cm/s, $a_{\max}^1$ is set to 5 cm/s², $a_{\min}^1$ is set to -5 cm/s², $a_{\max}^2$ is set to 5 cm/s², and $a_{\min}^2$ is set to -5 cm/s². Because we do not consider reversing in our experiment, $v_{\min}$ is set to 0 cm/s. In order to better observe the following state, we set τ to 0.1 ms. In order to ensure safety, we set D to 2 cm.

In general, when the distance between robot and target is less than the safe distance, we think that the state is in danger. Meanwhile, when the relative speed is equal to 0, the efficiency performance is thought of as well. Good stability performance requires that the acceleration of the robot following the target should be 0. Therefore, the average of $d - D$ and the frequency of $d < D$ can be used to evaluate the performance of safety and the StDev of $d - D$ can be used to evaluate the performance of efficiency. The average, StDev of $|\Delta v|$, and the frequency of $|\Delta v| \leq 0.1$ can also be used to evaluate the performance of efficiency. The average, StDev of $|a|$, and the frequency of $|a| = 0$ can be used to evaluate the performance of stability. Similarly, the average and StDev of $|\Delta a / \Delta t|$ can also be used to evaluate the performance of stability.

3.1. Training. The model needs to be trained first. The initial speed of the target is 0. During 0 to 30 seconds, it moves at constant acceleration which is 1 m/s². During 30 to 90 seconds, it moves at a constant speed. During 90 to 120 seconds, it moves at a changed speed that changes every 0.01 seconds and its value $\Delta v \sim U(-0.05, 0.05)$. After 210 seconds, the target slows down at constant acceleration, which

is -0.5 m/s² until the velocity becomes 0. The initial relative distance is 0, and the robot starts to follow the target 3 seconds after the target starts. The change of the velocity of the target in the training scene is indicated by a solid line in Figure 1.

After training, the policy of the model is basically stable.

3.2. Testing. The method of this paper was compared with the simulation method from paper 2, Q-learning method from paper 5, and DDPG algorithm from paper 6. The performance of the models was first tested in the training scene, and the data analysis of distance, velocity, and acceleration is shown in Table 1.

In order to distinguish the various algorithms, the policy of algorithms proposed in paper 2 is called $P1$, and the policy of algorithms proposed in paper 5 is called $P2$. Similarly, the policy of algorithms proposed in paper 6 is called $P3$. $P4$ and $P5$ are the policy proposed by this paper. $P4$ tends to get higher efficiency but $P5$ prefers getting higher stability. $P4$ and $P5$ are trained by the self-learning method but do not apply the multipolicy model. $P6$ adopts the multipolicy model, and its policy set includes $P4$ and $P5$, which prefer ensuring safety and efficiency when the target changes velocity frequently and perform stability when the target moves at a constant speed.

We processed the original data to some extent and obtained the average value, standard deviation, and frequency of corresponding data. The calculation formula is as

$$\text{average: } \mu = \frac{\sum_{i=1}^n x_i}{n}, \quad (9)$$

$$\text{StDev: } \sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2}, \quad (10)$$

$$\text{frequency: } f(x < k)\% = \frac{\sum_{i=1}^n |k - x| + (k - x)}{2(k - x) \cdot n}. \quad (11)$$

From Table 1, compared with $P1$ to $P3$, $P4$ to $P6$ perform better according to safety. Meanwhile, $P4$ and $P6$ also ensure high efficiency. $P5$ and $P6$ have excellent stability performance.

To verify the performance of the model in different environments, the following will test the performance in two different test scenarios:

Scene 1. The initial speed of the target is 0. During 0 to 15 seconds, it moves with an acceleration of 1.5 m/s² + Δa_{11} , $\Delta a_{11} \sim N(0, 0.5)$. During 15 to 60 seconds, it moves at a changed speed that changes every 0.01 seconds and its value $\Delta v \sim U(-0.1, 0.1)$. During 60 to 150 seconds, it moves at a changed speed that changes every 0.01 seconds and its value $\Delta v \sim U(-0.2, 0.2)$, and after 150 seconds, it moves with an acceleration of -0.5 m/s² until the speed is 0. The initial relative distance is 0, and the robot starts to follow the target 3 seconds after the target starts. The

FIGURE 1: Velocity of the target.

TABLE 1: The data analysis of the models in the training scene.

		$P1^3$	$P2^{10}$	$P3^{11}$	$P4$	$P5$	$P6$
$d - D$	Average	5.00	39.78	-0.55	0.73	35.72	3.55
	StDev	5.24	34.34	1.56	4.14	49.08	5.11
	$f(d < D)\%$	5.77	16.54	12.69	0.00	1.92	0.00
$ \Delta v $	Average	0.65	2.97	0.11	0.08	0.39	0.15
	StDev	0.51	1.65	0.33	0.08	0.48	0.25
	$f(\Delta v \leq 0.1)\%$	10.00	2.31	50.77	56.92	23.85	65.90
$ a $	Average	1.18	3.01	0.38	0.34	0.27	0.32
	StDev	0.82	1.46	0.24	0.17	0.19	0.21
	$f(a = 0)\%$	5.00	2.69	12.31	19.23	63.08	43.68
$ \Delta a / \Delta t $	Average	2.05	2.74	0.35	0.21	0.11	0.20
	StDev	1.72	1.84	0.67	0.38	0.36	0.41

$P1$ represents the policy of paper 3. $P2$ represents the policy of paper 10. $P3$ represents the policy of paper 11. $P4$ represents the policy of efficiency from this paper. $P5$ represents the policy of stability from this paper. $P6$ is an integrated model of $P4$ and $P5$.

change of the velocity of the target in scene 1 is indicated by a dashed line in Figure 1.

Scene 2. The initial speed of the target is 20 m/s. During 0 to 15 seconds, it moves at a constant speed. During 30 to 80 seconds, it moves with an acceleration of 0.1 m/s^2 . During 80 to 180 seconds, it moves at a constant speed. During 180 to 230 seconds it moves with an acceleration of -0.2 m/s^2 . When 230 to 250 seconds, it moves at a constant speed. The initial relative distance is 40, and the target and the robot begin to move at the same time. The speed variation of the target is indicated by a long-dashed line in Figure 1.

From Figure 1, we can find the difference between various methods. First, from the perspective of the change of the velocity of the target in the test scene, the moving speed of the target is changeable and irregular in scene 1 and the movement of the target is smooth and does not change greatly in scene 2. Obviously, the robot needs to ensure safety and improve efficiency as much as possible in scene 1. On the contrary, robots in scene 2 need not pay too much attention to the efficiency but need to improve the stability performance as much as possible.

From the results of the test scenarios on Tables 2 and 3, it is not difficult to find that $P1$ hardly keeps stable and sometimes does not leave enough distance with the target if its velocity changes fast. At the same time, $P2$ has bad stable performance and often cannot keep a safe distance with the target even if the target moves at a constant speed. $P3$ has excellent efficiency and not bad stable performance, but its poor security performance is worrying. $P4$ excessively pursues safety and efficiency without satisfactory stable performance. Similarly, $P5$ has excellent stable performance but poor efficiency. In terms of safety, efficiency, and stability, $P6$ mostly performed best in experiments.

From Figure 2, it is not difficult to find that these methods perform better in the training scene than in the testing scene, which proves that the new environment affects the performance of robot-following. Compared with other methods, the performance of $P4$, $P5$, and $P6$ is less affected. Therefore, the self-learning method has been proved to have better adaptability to the environment. At the same time, the efficiency performance of $P6$ is similar to that of $P4$, and the comfort performance of $P6$ is similar to that of $P5$, which shows that the multipolicy model can effectively improve the

TABLE 2: The data analysis of the models in test scene 1.

		$P1^3$	$P2^{10}$	$P3^{11}$	$P4$	$P5$	$P6$
$d - D$	Average	23.87	42.49	-0.34	3.29	38.29	3.99
	StDev	34.00	40.37	1.41	9.70	66.09	9.87
	$f(d < D)\%$	14.96	17.09	23.08	7.69	8.12	7.73
$ \Delta v $	Average	1.21	2.89	0.19	0.32	0.92	0.77
	StDev	1.08	1.77	0.26	0.62	1.27	0.98
	$f(\Delta v \leq 0.1)\%$	5.56	1.28	44.02	51.71	38.03	50.49
$ a $	Average	1.65	2.83	1.06	1.15	0.90	1.27
	StDev	1.10	1.47	0.70	0.74	0.73	0.81
	$f(a = 0)\%$	3.85	1.71	2.56	3.85	16.67	17.60
$ \Delta a / \Delta t $	Average	2.41	2.51	1.31	1.37	0.83	0.75
	StDev	1.80	1.77	1.57	1.75	1.15	1.09

$P1$ represents the policy of paper 3. $P2$ represents the policy of paper 10. $P3$ represents the policy of paper 11. $P4$ represents the policy of efficiency from this paper. $P5$ represents the policy of stability from this paper. $P6$ is an integrated model of $P4$ and $P5$.

TABLE 3: The data analysis of the models in the test scene 2.

		$P1^3$	$P2^{10}$	$P3^{11}$	$P4$	$P5$	$P6$
$d - D$	Average	21.06	54.63	10.13	16.09	35.72	17.70
	StDev	77.12	77.16	53.89	73.99	105.45	71.14
	$f(d < D)\%$	5.05	16.86	24.24	1.10	0.86	0.90
$ \Delta v $	Average	0.92	3.24	0.28	0.37	0.58	0.36
	StDev	1.66	1.99	1.54	1.64	1.78	1.64
	$f(\Delta v \leq 0.1)\%$	7.24	1.16	61.31	62.68	54.62	80.65
$ a $	Average	2.56	3.21	0.72	0.53	0.35	0.23
	StDev	1.48	1.59	0.67	0.64	0.63	0.44
	$f(a = 0)\%$	6.67	8.65	38.96	58.56	75.96	88.62
$ \Delta a / \Delta t $	Average	1.08	3.05	0.21	0.25	0.26	0.19
	StDev	0.93	1.45	0.71	0.77	0.76	0.74

$P1$ represents the policy of paper 3. $P2$ represents the policy of paper 10. $P3$ represents the policy of paper 11. $P4$ represents the policy of efficiency from this paper. $P5$ represents the policy of stability from this paper. $P6$ is an integrated model of $P4$ and $P5$.

FIGURE 2: Continued.

FIGURE 2: Performance of various models. (a) Safety. (b) Efficiency. (c) Stability.

comprehensive performance of robot-following and better complete the following task.

To sum up, the multipolicy model overcomes the shortcoming that the traditional single-strategy model cannot adapt to different environments. By using different policies according to different environments, the multipolicy model can perfectly meet people's different requirements for different environments. Therefore, it has high practicability and the value of the study.

4. Conclusion

In this paper, a Markov decision model for the robot-following process is first modeled, then a multiobjective reinforcement learning algorithm is proposed by improving the existing reinforcement learning algorithm, and the agent could obtain a multipolicy model.

This method improves the stability performance by 6 times without affecting the robot-following efficiency when the speed of the target changes frequently and the stability performance can be improved by 2 times when the target moves smoothly.

In summary, we adopt a multipolicy model, which can take different policy preferences for different environments, so as to ensure that the optimal decision can be made in most cases. Meanwhile, we adopt the self-learning method to avoid the problem of artificial weighting, so that the robot can determine the weight ratio by itself and realize the intelligence of the robot to a greater extent. In addition, we introduce more objective considerations into the algorithm, so that the final model is a multiobjective and multipolicy agent. The contributions of this paper mainly include the following:

- (1) Self-learning mode is proposed to solve the problem that the reward function is difficult to determine in the robot-following problem.
- (2) The adoption of pilot experience structure reduces a large number of unnecessary explorations in the early stage of intensive learning and training.

- (3) The multipolicy model can effectively improve the adaptability of the algorithm to the environment so that the decision can meet the needs of more users.

However, because the algorithm is currently only tested in simulation and real experiment has not been done yet, there are still many unknown factors influencing the actual environment. Further experiments in real environment are needed.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The work was supposed by the National Key Research and Development Program of China under Grants 2020YFB1708503, 2019YFB1705003, and 2018YFB1308801.

References

- [1] C. C. Chien and P. Ioannou, "Automatic vehicle-following," in *Proceedings of the American Control Conference*, pp. 1748–1752, IEEE, Chicago, IL, USA, Junen 1992.
- [2] J. M. Kuefler, T. Wheeler, and M. Kochenderfer, "Imitating driver behavior with generative adversarial networks," in *Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 204–211, IEEE, Los Angeles, CA, USA, June-2017.
- [3] Z. Gao, W. Yan, H. Li, and Z. Hu, "A vehicle adaptive cruise control algorithm based on simulating driver's multi-objective decision making," *Automotive Engineering*, vol. 37, no. 6, pp. 667–673, 2015.
- [4] R. Li, L. Zhang, and J. Wang, "A control method of unmanned car following under time-varying relative distance and angle," *Acta Automatica Sinica*, vol. 48, no. 11, pp. 2031–2040, 2018.

- [5] R. Dechter, "Learning while searching in constraint-satisfaction problems," in *Proceedings of the University of California, Computer Science Department, Cognitive Systems Laboratory*, Philadelphia, PA, USA, August 1986.
- [6] Y. Gao, S. Chen, and X. Lu, "The review of reinforcement learning," *Acta Automatica Sinica*, vol. 30, no. 1, pp. 86–100, 2004.
- [7] X. Wang, R. Jiang, L. Li, Y. Lin, X. Zheng, and F.-Y. Wang, "Capturing car-following behaviors by deep learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 910–920, 2018.
- [8] M. Awad and R. Khanna, "Deep neural networks," *Efficient Learning Machines*, Apress, Berkeley, CA, USA, 2015.
- [9] W. Yang, C. Bai, C. Cai, Y. Zhao, and P. Liu, "Survey on sparse reward in deep reinforcement learning," *Computer Science*, vol. 47, no. 3, pp. 182–191, 2020.
- [10] Z. Gao, T. Sun, and H. Xiao, "Decision-making method for vehicle longitudinal automatic driving based on reinforcement Q-learning," *International Journal of Advanced Robotic Systems*, vol. 16, no. 3, pp. 1–13, 2019.
- [11] X. Deng, J. Hou, G. Tan, B. Wan, and T. Cao, "Multi-objective vehicle following decision algorithm based on reinforcement learning," *Control and Decision*, vol. 36, no. 6, pp. 1–7, 2020.
- [12] H. Gao, G. Shi, K. Xie, and Y. Liu, "Research on decision-making of autonomous vehicle following based on reinforcement learning method," *Industrial Robot: The International Journal of Robotics Research and Application*, vol. 46, no. 3, pp. 444–452, 2019.
- [13] H. Gao, G. Shi, G. Xie, and B. Cheng, "Car-following method based on inverse reinforcement learning for autonomous vehicle decision-making," *International Journal of Advanced Robotic Systems*, vol. 15, no. 6, pp. 1–11, 2019.
- [14] R. S. Sutton, D. A. Mcallester, S. P. Singh, and M. Yishay, "Policy gradient methods for reinforcement learning with function approximation," in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, MIT Press, Cambridge, MA, USA, November 1999.
- [15] J. Peters and S. Schaal, "Natural actor-critic," *Neurocomputing*, vol. 71, no. 7–9, pp. 1180–1190, 2008.
- [16] B. Wu and Y. Feng, "Policy reuse for learning and planning in partially observable Markov decision processes," *International conference on information science and control engineering*, vol. 4, pp. 549–552, 2017.
- [17] J. Tong, X. Xu, L. Hou, Y. Li, J. Wang, and L. Zhang, "An ultra-short baseline positioning model based on rotating array & reusing elements and its error analysis," *Sensors*, vol. 19, no. 20, pp. 1–23, 2019.