Research
A SCIENCE PARTNER JOURNAL

RESEARCH ARTICLE

Quantitative Representation of Autonomous Driving Scenario Difficulty Based on Adversarial Policy Search

Shuo Yang[1], Caojun Wang[1], Yuanjian Zhang[2], Yuming Yin[3], Yanjun Huang[1,4*], Shengbo Eben Li[5], and Hong Chen[6]

[1]School of Automotive Studies, Tongji University, Shanghai 201804, China. [2]Department of Aeronautical and Automotive Engineering, Loughborough University, Loughborough, Leicestershire LE11 3TU, UK. [3]School of Mechanical Engineering, Zhejiang University of Technology, Hangzhou 310014, China. [4]Frontiers Science Center for Intelligent Autonomous Systems, Shanghai 200120, China. [5]School of Vehicle and Mobility, Tsinghua University, Beijing 100084, China. [6]College of Electronics and Information Engineering, Tongji University, Shanghai 201804, China.

*Address correspondence to: yanjun_huang@tongji.edu.cn

Citation: Yang S, Wang C, Zhang Y, Yin Y, Huang Y, Li SE, Chen H. Quantitative Representation of Autonomous Driving Scenario Difficulty Based on Adversarial Policy Search. *Research* 2025;8:Article 0575. https://doi.org/10.34133/research.0575

Submitted 13 March 2024
Revised 16 December 2024
Accepted 17 December 2024
Published 17 January 2025

Autonomous vehicles with self-evolution capabilities are expected to improve their performance through learning algorithms, to automatically adapt to the external environment. However, due to the infinity, complexity, and variability of the actual traffic environment, it is necessary to develop quantitative representation indicators of scenario difficulty and generate targeted scenarios to ensure the evolution gradually, so as to quickly approach the performance limit of the algorithm. Therefore, this paper proposes a data-driven quantitative representation method of scenario difficulty. Specifically, the concept of environment agent is proposed, and a reinforcement learning method combined with mechanism knowledge is constructed for policy search to obtain an agent with an adversarial behavior. The model parameters of the environment agent at different stages in the training process are extracted to construct a policy group, and then agents with different adversarial intensities are obtained, which are used to realize data generation in different difficulty scenarios through the simulation environment. Finally, a data-driven scenario difficulty quantitative representation model is constructed, which is used to output the environment agent policy under different difficulties. Experimental results show the effectiveness of the proposed method. The result analysis shows that the proposed algorithm can generate reasonable and interpretable scenarios with high discrimination and can provide quantifiable difficulty representation without any expert logic rule design. Compared with the rule-based discrete scenario difficulty representation method, the proposed algorithm can achieve continuous difficulty representation. The video link is https://www.youtube.com/watch?v=GceGdqAm9Ys.

Introduction

Autonomous driving technology has received more and more attention with the increasing level of automobile intelligence [1]. However, due to the challenges of algorithm level and technology maturity, there is still a distance to the realization of high-level autonomous driving. In recent years, self-evolutionary algorithms with experience storage and learning upgrade as the core idea have become a research hotspot [2–4]. Similar to human beings learning driving skills from limited scenarios to cope with infinite scenarios, autonomous vehicles with self-evolutionary algorithms are considered to have the potential to adapt to a huge number of scenarios in the real world and to realize a gradual improvement in performance [5–7].

Autonomous driving algorithms are required to undergo tens of millions of miles of data collection and testing before they can be deployed in real-world applications [8,9]. Self-evolving autonomous vehicles need to face enough scenarios and expose the algorithm's problems, so this process is considered fundamental for the algorithm to achieve continuous self-improvement. However, the actual traffic environment is complex and it is difficult to be exhaustive, so it is necessary to obtain diverse, typical, reasonable, and interpretable environmental input information through scenario generation methods, so as to accelerate the efficiency of self-evolution and test evaluation [10,11]. There have been a lot of studies on related techniques, mainly including mechanism model methods and data-driven methods.

Scenario generation based on mechanistic models aims to generate and simulate scenarios that may be encountered by autonomous vehicles by utilizing experts' understanding and professional experience of the changing rules within the scenario system, combined with logic rules and optimized solving. Rocklage et al. [12] proposed a method for automatically generating autonomous driving scenarios for regression testing. The method defines different types of traffic scenarios by creating

static and mixed scenarios to guarantee a certain coverage of parameter combinations. de Gelder et al. proposed a scenario parameter generation method and a scenario representativeness metric. This method can determine enough parameters for scenario description and generate real parameter values by estimating the probability density function of these parameters. A Wasserstein-distance-based scenario representativeness metric is also proposed to quantify the realism of the generated scenario [13].

However, although mechanism model-based methods can make full use of existing knowledge to generate diverse and interpretable scenarios according to the test requirements, the artificial modeling method relies on a large number of expert experiences and various simplified conditions, which limits the practical application of these techniques. Compared with mechanistic modeling methods, data-driven methods have an obvious advantage because they can mine the potential laws from the data and fully consider the nonlinearities, uncertainties, and characteristic probability distributions in scenario generation [14,15].

Data-driven methods can be mainly divided into 2 categories, namely, natural scenario generation and adversarial scenario generation. Natural scenario generation aims to train a scenario generation model that is consistent with real driving conditions based on massive natural driving data through Bayesian networks [16], deep learning [17], and other methods. Diverse scenarios are generated by data-driven methods for training and testing of autonomous driving algorithms [18].

However, in natural scenarios, the probability of accidents for autonomous driving algorithms is very small, so it is necessary to construct targeted scenarios through adversarial scenario generation methods to approach the performance limit of algorithms quickly [19–21]. Zhang et al. [22] proposed a metamorphic testing framework for autonomous driving systems based on a generative adversarial network, which is combined with real-world weather scenarios to generate driving scenarios under various extreme weather conditions. Jia et al. [23] proposed a dynamic scenario generation method based on conditional generative adversarial imitation learning, in which scenario class labels are incorporated into the model to generate different types of traffic scenarios for inferring the weaknesses of the under-test autonomous vehicles. Rempe et al. [24] proposed STRIVE, a method that uses a learned traffic model to automatically generate realistic, challenging traffic scenarios that expose weaknesses in autonomous vehicle planners, ultimately enabling the improvement of these planners through targeted scenario generation. Hao et al. [25] proposed an adversarial safety-critical scenario generation method based on natural human driving priors, which uses human driving priors and reinforcement learning (RL) techniques to generate realistic safety-critical test scenarios covering both naturalness and adversarial.

It can be seen that the adversarial scenario generation method based on data-driven method can more effectively find the vulnerabilities of the algorithm and greatly accelerate the efficiency of algorithm training and testing. However, due to the black-box nature of data-driven methods, there are few studies that can generate interpretable and targeted quantifiable scenarios with different difficulty levels through explicit quantitative indicators of scene difficulty. However, the black-box nature of data-driven algorithms makes it difficult to accurately quantify and interpret the results of generated scenarios. This difficulty has led to few studies that clearly define quantitative criteria for scenario difficulty in order to generate interpretable and targeted quantifiable scenarios with different difficulty levels through explicit quantitative indicators of scenario difficulty.

In view of the above problems, this paper proposes a data-driven quantitative representation method of scenario difficulty for autonomous driving based on environment agent policy search. To our knowledge, this is the first proposed data-driven approach for quantifying scenario difficulty representation. The concept of environment agent is proposed, and an RL algorithm combined with mechanism knowledge is constructed to realize the policy search, so as to obtain the agent policy with an adversarial behavior. To obtain information on the quantitative dimension of scenario difficulty, the model parameters of environment agents at different stages of the training process are extracted into adversarial policy groups to obtain agent policies with different adversarial intensities. In the simulation environment, the data generation of the scenario with different difficulty levels is carried out and a scenario database is constructed. A data-driven scenario difficulty quantitative representation model is constructed, and the feature correlation of scenario input information is extracted through the attention network, and finally an environment agent policy that can output different difficulty scenarios is obtained. The proposed method can generate reasonable scenarios with high discrimination and can provide quantifiable difficulty representation without any expert logic rule design.

The contributions of this study are summarized as follows:

- This paper proposes the concept of environment agent, combines mechanism knowledge and the RL method to achieve efficient policy search, and obtains agent policies with adversarial behaviors.
- This paper proposes a data generation method for varying difficulty scenarios, which combines the policy groups constructed by model parameters at different stages in the training process to provide information on the quantitative dimension of scenario difficulty.
- This paper proposes a data-driven scenario difficulty quantitative representation model and proves that it can generate highly distinguishable scenarios with reasonable and quantifiable difficulty representations through result analysis.

This paper is organized as follows: The proposed framework is introduced in the "Proposed framework" section. The environment agent policy search is introduced in the "Environment agent policy search" section. The descriptions of data generation of varied difficulty scenarios are proposed in the "Data generation of varied difficulty scenarios" section. The details of quantitative representation of scenario difficulty are proposed in the "Experiment setting" section. In the "Simulation results" section, our method is verified and compared in simulation, and Conclusion concludes this paper.

Results and Discussion

Experiment setting

In this section, the proposed data-driven quantitative representation method of scenario difficulty for autonomous driving is validated in a simulation platform. The training scenario consists of randomly generated traffic flows with speeds ranging from 8 to 12 m/s within 180 m of the ego vehicle, as well as the

surrounding vehicle with an adversarial behavior defined as the environmental agent. The training environment is constructed in the simulation software CARLA [26].

The soft actor algorithm consists of the value network and the policy network. The value network is a fully connected neural network with 3 layers, including 5 input neurons, 1 output neuron, and 256 hidden layer neurons. The policy network is a 4-layer fully connected neural network with 5 input neurons, 2 output neurons, and 256 hidden layer neurons. The scenario difficulty quantitative representation model φ consists of a transform decoder and a fully connected layer of 7 layers, where the number of neurons in the fully connected layer is 512.

The parameters of the algorithm are set as shown in Table 1. The parameter settings are subject to detailed selection and tuning. The simulation step size $\Delta t = 0.1$ s balances computational efficiency and simulation accuracy; the value network discount factor $\gamma = 0.99$ ensures that future rewards are not overly discounted, promoting long-term decision-making; the learning rate of the adjust temperature $\lambda = 3 \times 10^{-4}$ and the learning rate of the target network $\tau_1 = 5 \times 10^{-3}$ are fine-tuned through preliminary experiments to ensure stability and efficient convergence; the total number of training steps $n_t = 100,000$ is determined based on the task complexity and observed convergence behavior; and the sampling interval $k = 10$ balances the amount of training data used and computational efficiency.

Simulation results

The algorithm is deployed in the training environment. The computer is equipped with an Intel Core i7-10700 central processing unit and an NVIDIA GeForce GTX 1660 SUPER graphics processing unit. The ego vehicle's policy is chosen as the learning-based autonomous driving algorithm with efficient self-evolutionary capabilities [3]. Specifically, the algorithm used is the advanced given algorithm accelerate-soft actor critic method, which combines behavioral cloning (BC) and soft actor critic (SAC) RL to optimize both efficiency and adaptability in complex traffic environments. This approach allows the ego vehicle to evolve autonomously by leveraging a mixture of suboptimal policy guidance and dynamic policy exploration, ensuring both accelerated convergence and enhanced adaptability to challenging driving scenarios.

Table 1. Hyperparameters for the simulation

Hyperparameters for the simulation	Symbol and value
Simulation step size	$\Delta t = 0.1$ s
Discount factor of the value network	$\gamma = 0.99$
Learning rate of the adjust temperature	$\lambda = 3 \times 10^{-4}$
Learning rate of the target network	$\tau_1 = 5 \times 10^{-3}$
Total training steps	$n_t = 100,000$
Sampling interval	$k = 10$
Same category data size	$n_{sc} = 10,000$
Episode number	$n_{episode} = 1,000$
Batch size	$n_b = 1,024$
Epoch number	$n_{epoch} = 1,000$
Lane width	$L_{lane} = 3.5$ m

Quantitative results

The effectiveness of the scenario difficulty generation method proposed in this paper is analyzed through quantitative results.

Figure 1 shows the trajectories, velocities, and lateral displacements of the ego vehicle and the environment vehicles under various adversarial scenarios of different intensities. Specifically, Fig. 1A to C present the results when the scenario difficulty factor X_{scd} is 0.2, 0.5, and 0.8, respectively. When the scenario difficulty factor is low, as in Fig. 1A, the speed change of the environment agent is smooth, resulting in little effect on the ego vehicle following behind. When the scenario difficulty factor $X_{scd} = 0.5$, as in Fig. 1B, the environment agent produces a more pronounced adversarial behavior. This is evident from the increased lateral offset, as it cuts in more aggressively while also generating sharp acceleration and deceleration to minimize the performance of the ego vehicle. In the most challenging scenario, the environment agent is able to actively resist the ego vehicle by jointly controlling the lateral offset and longitudinal speed until a collision occurs.

Figure 2 shows the trajectory, scenario difficulty, velocity, and lateral displacement of the ego vehicle and the environment vehicles under a continuously varying scenario difficulty factor. The experiment is divided into 3 stages. In the first stage (stage 1), the scenario difficulty $X_{scd} = 0.13$. At this stage, the behavior of surrounding vehicles is relatively calm, having minimal impact on the ego vehicle, and the vehicle speed is relatively stable, indicating that the behavior generated under low-difficulty scenarios is reasonable and smooth. In the second stage (stage 2), the scenario difficulty gradually increases and then decreases. Surrounding vehicles periodically accelerate and decelerate to confront the ego vehicle, causing fluctuations in vehicle speed. In the third stage (stage 3), the scenario difficulty is raised to 1, reaching the maximum difficulty. The adversarial behavior of the surrounding vehicles is most intense at this stage, ultimately leading to a collision between the ego vehicle and the surrounding vehicles, verifying the reasonableness of the extreme behavior generated under high-difficulty confrontation scenarios.

Qualitative results

Through the qualitative results, the reasonability of the proposed method can be further analyzed.

The visualization results of quantitative representation of scenario difficulty is shown in Fig. 3. Figure 3A to C illustrate the contribution ordering among input states for the fixed scene difficulty factor ($X_{scd} = 0.2,0.5,0.8$). It can be seen that as the scenario evolves, the change in the contribution ordering of state S consists of 3 phases:

1. Overtaking phase: the environment agent accelerates in order to reach the front of the ego vehicle to ensure that it can have an opportunity to influence the vehicle. At this point, state Δs has the highest contribution.
2. Cut-in phase: the environment agent in front of the ego vehicle generates adversarial behavior as much as possible through the control of the longitudinal movement. At this point, states v_s and a_s have the highest contribution.
3. Maintenance phase: the environment agent is in the process of confrontation with the ego vehicle. If there is any lateral offset deviation between the environment agent and the ego vehicle, the environment agent must block the ego vehicle and continue to produce adversarial behaviors. At this stage, state Δd has the highest contribution.

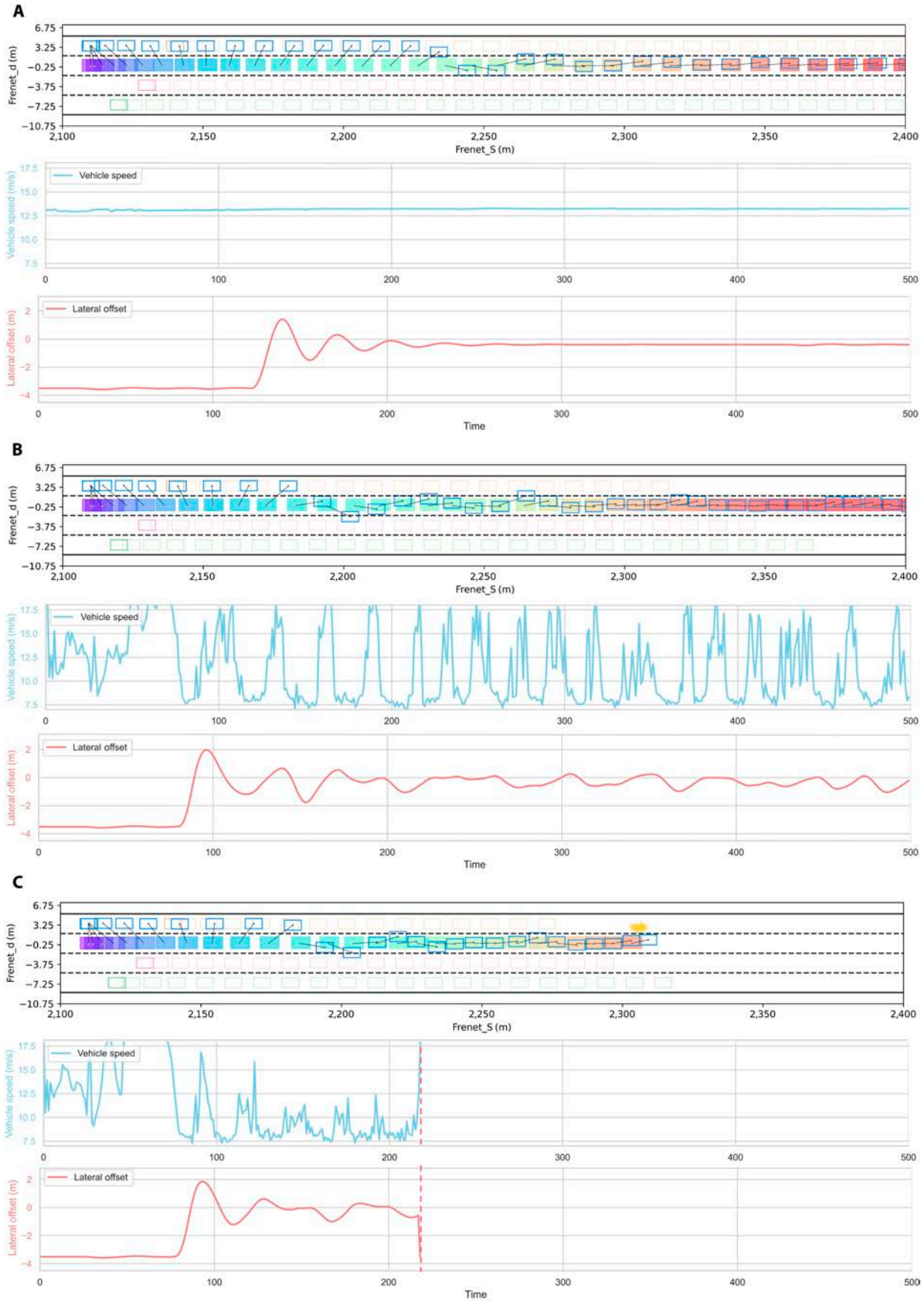

Fig. 1. The trajectories, velocities and lateral displacements of the ego vehicle and the environment vehicles under various adversarial scenarios of different intensities. (A) Scenario difficulty factor $X_{scd} = 0.2$. (B) Scenario difficulty factor $X_{scd} = 0.5$. (C) Scenario difficulty factor $X_{scd} = 0.8$.

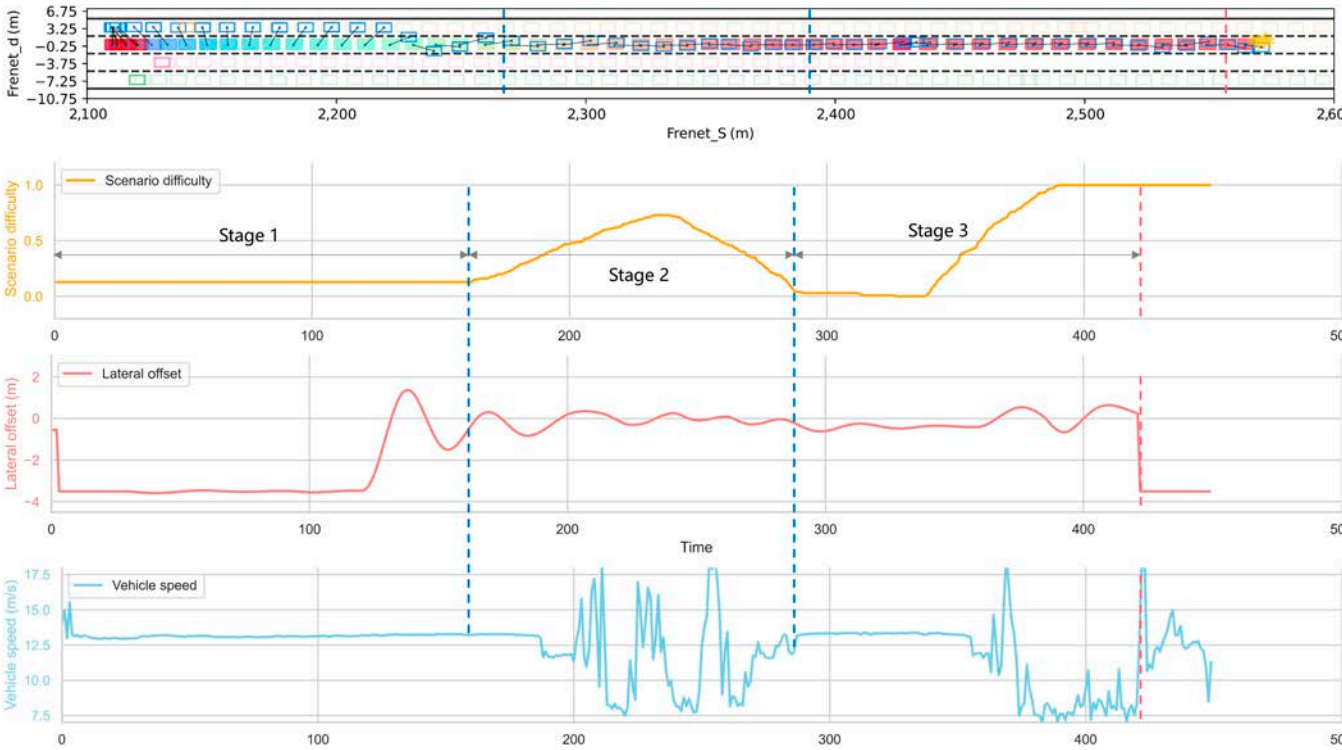

Fig. 2. The trajectory, scenario difficulty, velocity, and lateral displacement of the ego vehicle and the environment vehicles under a continuously varying scenario difficulty factor.

The above analysis process illustrates that the proposed data-driven quantitative representation method of scenario difficulty not only realizes the generation of variable difficulty scenarios but also has a certain level of interpretability compared with traditional black-box methods.

Figure 3D illustrates a scenario difficulty factor X_{scd} that dynamically changes with manual input. It can be seen that the proposed data-driven model can generate adversarial scenarios with continuous variable difficulty, and the correlation degree between states is also interpretable.

Figure 3E shows the quantization results of the scenario difficulty. The proposed scenario difficulty quantitative representation model is deployed to generate variable difficulty scenarios. The average reward for each episode is saved and used as a quantitative metric. It can be seen that the proposed method can generate diverse scenarios with quantifiable difficulty.

t-distributed stochastic neighbor embedding (t-SNE) is a machine learning algorithm for dimensionality reduction and visualization of high-dimensional data [27]. This method can be used to understand and present patterns and relationships in high-dimensional data. In this paper, the t-SNE method is applied to process the data in the varied difficulty scenario dataset. The t-SNE visualization of varied difficulty scenarios is shown in Fig. 3F. It can be seen that the scenario data of different difficulty levels have obvious aggregation. This shows that the proposed method can generate variable difficulty adversarial scenarios with discrimination.

Conclusion

This paper proposes a data-driven quantitative representation method of scenario difficulty. The method constructed a variable-difficulty scenario generation model based on the transformer architecture through 2 key steps, namely, the policy search of the environment agent and the data generation of varied difficulty scenarios. The method was validated in a typical adversarial scenario. The experimental results demonstrated that the proposed algorithm can generate reasonable and highly distinguishable scenarios with quantifiable difficulty representations without any expert logic rule design. This applied to both fixed and dynamically changing scenario difficulty factor inputs. Furthermore, the analysis results indicated that the proposed method has a certain level of interpretability compared to traditional black-box methods. Future research will include large-scale natural driving datasets to generate more realistic adversarial scenarios. Additionally, the proposed method will be extended to cover more adversarial scenario generation tasks.

Methods

Proposed framework

As illustrated in Fig. 4, the proposed method comprises 3 main components: environment agent policy search, data generation for scenarios of varying difficulty, and training of a data-driven model to quantify scenario difficulty. First, environment agent policy search leverages RL to update policies and generate adversarial behaviors. Next, scenarios of varying difficulty are generated using models from different stages of training. Finally, a quantitative representation model is trained to extract feature associations from the scenario input data. The model inputs include the traffic environment information and a continuous scenario difficulty value, and the output is the action of the environment agent.

This method employs a deep neural network as the environment agent and addresses the problem of adversarial policy search through an RL algorithm. As shown in Fig. 5, the model parameters trained at various stages are updated, saved, and output into the constructed policy group. The

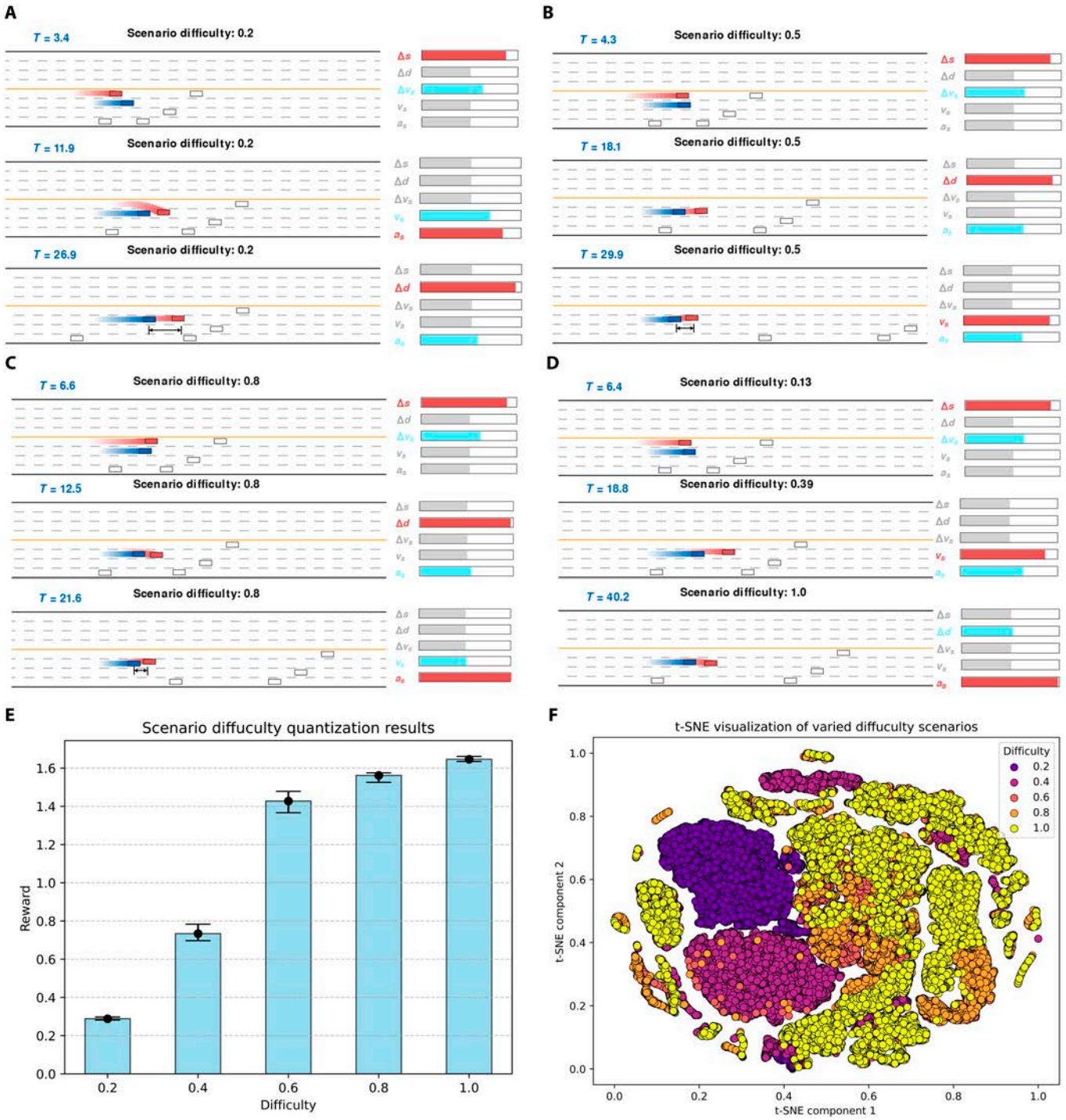

Fig. 3. Visualization results of quantitative representation of scenario difficulty. (A) Scenario difficulty factor $X_{scd} = 0.2$, showing a moderate adversarial behavior. (B) Scenario difficulty factor $X_{scd} = 0.5$, showing an increased adversarial intensity closer to the ego vehicle's limits. (C) Scenario difficulty factor $X_{scd} = 0.8$, showing an increased adversarial intensity closer to the ego vehicle's limits. (D) Scenario difficulty factor X_{scd} that dynamically changes with manual input. (E) Scenario difficulty quantitation results. (F) t-distributed stochastic neighbor embedding (t-SNE) visualization of varied difficulty scenarios.

process of data generation varying difficulty scenarios involves 2 parts: deploying adversarial policies from the policy group and constructing the scenario database. Multiple policies from the policy group are successively deployed to the simulation environment for data generation of varying difficulty scenarios and appended to the scenario database. The proposed method mainly includes 2 phases: the training phase and deployment phase. In the training phase, the attention mechanism model is used to extract the feature association between the scenario difficulty factor and the state input. This allows for the creation of an environment agent policy capable of outputting different difficulty scenarios. Furthermore, in the deployment phase, the environment agent is deployed to generate scenarios with continuous difficulty levels.

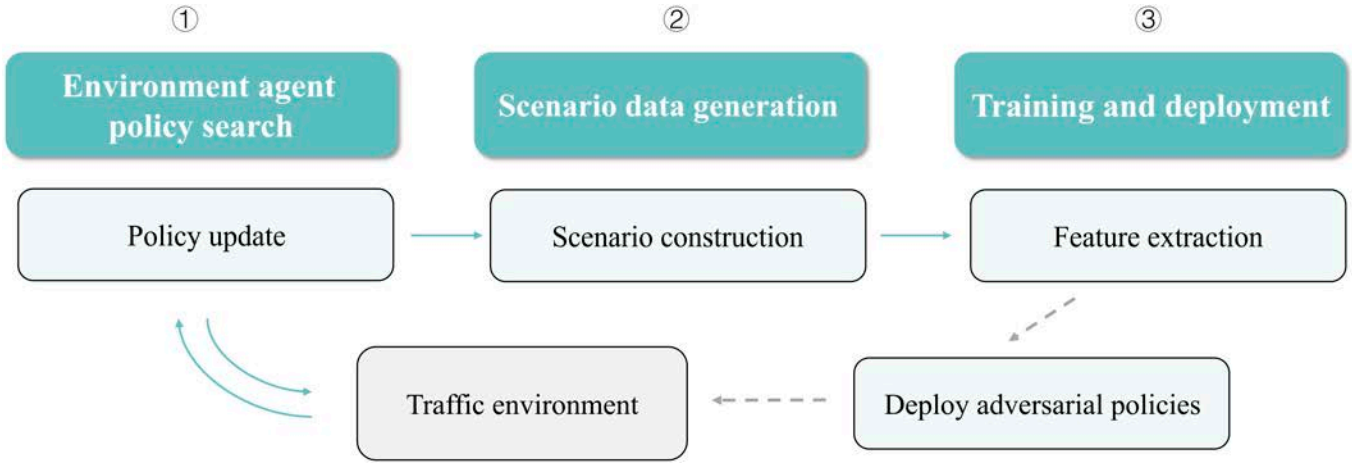

Fig. 4. Overall architecture of the data-driven quantitative representation method of scenario difficulty.

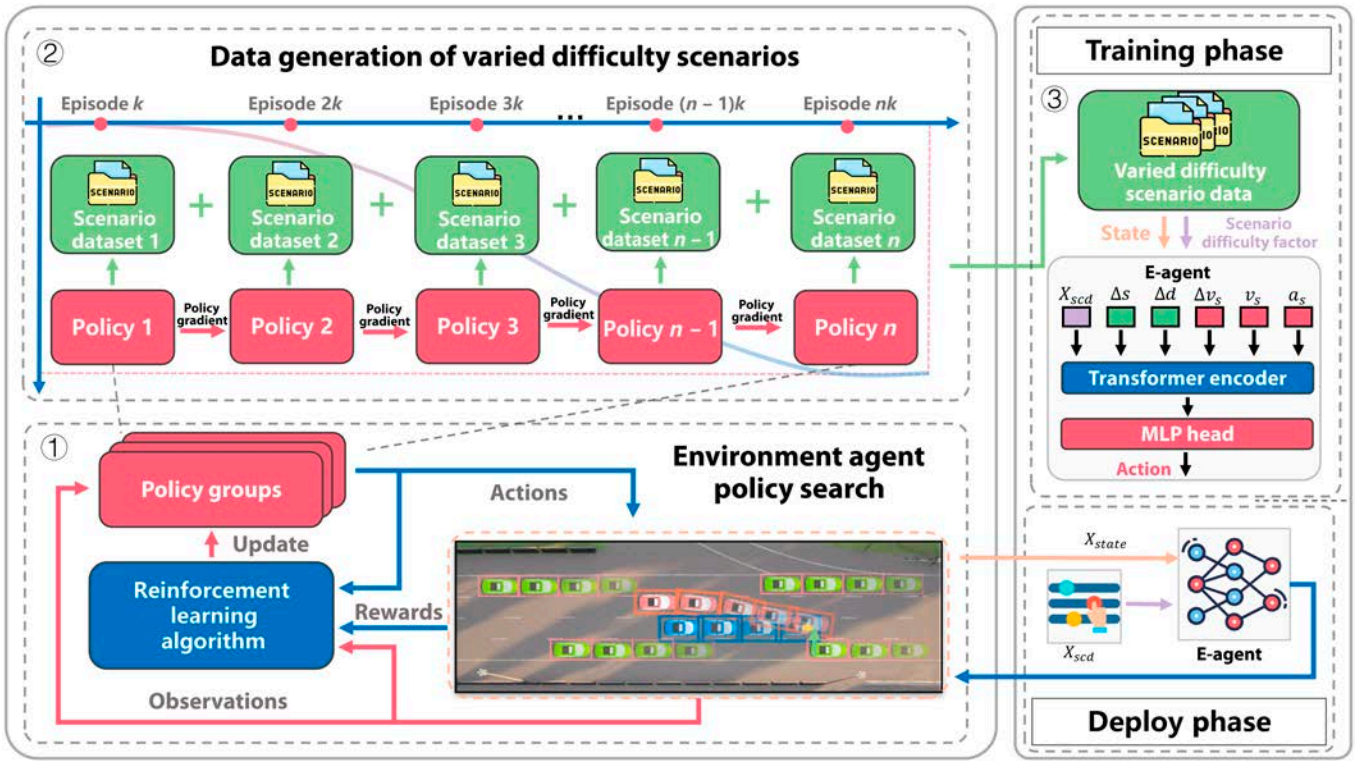

Fig. 5. A detailed process diagram of the data-driven scenario difficulty quantitative representation method, including environment agent policy search, data generation of varied difficulty scenarios, and training and deployment phases of the scenario difficulty model. The process starts with policy search, generates scenario datasets of different difficulty levels, and then trains a quantitative representation model to extract scenario features and deploys the environment agent to create scenarios with continuous difficulty levels. MLP, multilayer perceptron.

Environment agent policy search

Problem definition

Adversarial scenario generation aims to construct various complex, dangerous, or extreme traffic scenarios in order to expose the weaknesses of autonomous driving algorithms when facing different environmental inputs. Based on this, improving the performance of the algorithms based on the evaluation results can further improve the reliability and safety of the autonomous driving system.

In this paper, the concept of environment agent is presented for generating adversarial scenarios. The optimization objective of the environment agent policy is to maximize the influence of the scenario information input on the performance of the ego vehicle through the agent policy search. The optimization problem can be formulated as follows:

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\left[F\left(\pi_A,\pi_E(\theta),S_c\right)\right], \tag{1}$$

where $F(\cdot)$ is the performance evaluation metric, π_A is the ego vehicle policy, π_E is the environment agent policy with parameter θ, and S_c is the scenario dynamics. S_c is used to characterize the state transitions of all agents.

The RL algorithm is dedicated to searching for optimal policies in order to maximize future returns. Therefore, this paper constructs an RL problem to realize the policy search for the above optimization problem. Considering that the adversarial behaviors of the environment agent must satisfy the basic logic rules, this paper incorporates the mechanism knowledge in order to speed up the efficiency of the policy search.

Environment agent design

Markov decision process

The dynamic interaction process between the environment agent and the ego vehicle is constructed as a Markov decision process (MDP), which can be defined by the following 5 tuples:

$$M = (S, A, P, \pi, R), \tag{2}$$

where S is the state space, A is the action space, P is the probabilistic model of state transition, π is the policy model, and R is the reward.

The core content of the MDP is the Markov property. It states that the future state of a system depends solely on its current state, independent of the sequence of states that led to the current state. In other words, the next state in the sequence is completely determined by the current state and is not affected by the sequence of previous states. The Markov property can be expressed as

$$\begin{aligned} p(s_{t+1}|s_t) &= p(s_{t+1}|h_t), \\ p(s_{t+1}|s_t, a_t) &= p(s_{t+1}|h_t, a_t), \end{aligned} \tag{3}$$

where the history of states is $h_t = \{s_1, s_2, s_3, \ldots, s_t\}$.

Soft actor critic

RL is a powerful tool for solving MDPs and find optimal or suboptimal policies for learning agents. The SAC algorithm [28] is a widely used RL algorithm. It is a model-free, off-policy algorithm based on an actor-critic framework. SAC uses stochastic policy, which can make the policy as random as possible. The agent can explore the state space S more fully, avoid the policy falling into the local optimum early, and can explore multiple feasible solutions to complete the specified task to improve anti-interference ability. It should be noted that the proposed algorithm is not limited to the use of the SAC algorithm, but other novel RL methods can be also applied to the method.

According to MDP, SAC seeks to solve the following maximum entropy problem:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \underset{(s_t,a_t)\sim\rho_\pi}{} \left[\sum_t r(s_t, a_t) + \alpha \mathcal{H}\left(\pi(\cdot|s_t)\right) \right], \tag{4}$$

where $\mathcal{H}$ is entropy and α is the temperature parameter. With $\alpha \to 0$, the maximum entropy RL gradually approaches the conventional RL.

The basic flow of soft policy iteration is to run 2 steps of policy evaluation and policy improvement alternately until convergence. In the policy evaluation step of soft policy iteration, the value of policy π is calculated according to the maximum entropy objective. The soft Q value can be obtained by iterating the Behrman backup operator, that is,

$$\mathcal{T}^\pi Q(s_t, a_t) \triangleq r(s_t, a_t) + \gamma_{\mathbf{s}_{t+1}\sim p}\left[V(s_{t+1})\right] \tag{5}$$

The goal of soft policy improvement is to search for a new policy π_{new} that is better than the current policy π_{old}. To achieve this goal, we can represent the policy as a Gaussian distribution and reduce the gap between the current policy and the new policy by minimizing the Kullback–Leibler divergence.

$$\pi_{\text{new}} = \arg\min_{\pi' \in \Pi} D_{\text{KL}}\left(\pi'(\cdot|\mathbf{s}_t) \,\|\, \frac{\exp\left(Q^{\pi_{\text{old}}}(\mathbf{s}_t, \cdot)\right)}{Z^{\pi_{\text{old}}}(\mathbf{s}_t)} \right), \tag{6}$$

where $Z^{\pi_{\text{old}}}(\mathbf{s}_t)$ is the normalized distribution of Q values; it will not contribute to the policy gradient, so it can be ignored.

Soft policy iteration has convergence and optimality; see Haarnoja et al. [29].

State and action space design

During the normal operation of autonomous vehicles, a sudden cut-in of environment vehicles is a common risky behavior that may lead to emergency situations and jeopardize traffic safety. First, a sudden cut-in may disrupt the autonomous vehicle's path and speed planning, requiring the system to react quickly to avoid collisions or violations. Second, sudden engagement can trigger emergency braking or evasive maneuvers, and the braking system and evasive strategies need time to respond, potentially resulting in collisions with rear-end vehicles or roadway blockages that can lead to traffic congestion and more serious accidents.

Figure 6 presents a schematic of the environment agent generating an adversarial behavior. Among them, the blue car represents the ego vehicle and the red car represents the environment agent. When the ego vehicle is driving normally, the environment agent will look for a time to cut in from the neighboring lanes to the current lane and minimize the ego vehicle's performance as much as possible through the control of the throttle, brake, and steering, so as to maximize the risk of side collision, corner collision, or front collision.

The state design is required to fully consider the input information required by the environment agent. Define the state space S as follows:

$$S = \begin{bmatrix} \Delta s & \Delta d & \Delta v_s & v_s & a_s \end{bmatrix}, \tag{7}$$

where Δs is the relative longitudinal distance between the environment agent and the ego vehicle, Δd is the relative lateral distance, Δv_s is the relative longitudinal velocity, and v_s and a_s are the longitudinal velocity and the longitudinal acceleration of the environment agent, respectively.

The underlying principle of the adversarial policy generated by the environment agent is that the agent should be as close to the ego vehicle as possible to maximize the risk of collision. Therefore, the behavior toward the environment agent moving away from the ego vehicle is against the basic rules of logic. To solve this problem, this paper combines the trajectory planning method based on mechanism rules to construct the RL problem and the action space.

As shown in Fig. 7, the quintic and quartic polynomial curves are respectively applied to describe the longitudinal and lateral trajectory planning process of the environment agent [30]:

Fig. 6. Adversarial behavior of the environment agent. The blue car represents the ego vehicle driving normally, and the red car represents the environment agent with adversarial behavior.

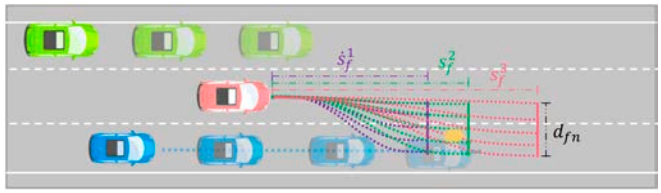

Fig. 7. Construction of the adversarial policy search problem for the environment agent based on polynomial curves.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 1 & t & t^2 & t^3 & t^4 & t^5 \\ 0 & 1 & 2t & 3t^2 & 4t^3 & 5t^4 \\ 0 & 0 & 2 & 6t & 12t^2 & 20t^3 \end{bmatrix} \cdot p_d = \begin{bmatrix} d_0 \\ \dot{d}_0 \\ \ddot{d}_0 \\ d_{fn} \\ \dot{d}_{fn} \\ \ddot{d}_{fn} \end{bmatrix}, \tag{8}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 1 & 2t & 3t^2 & 4t^3 \\ 0 & 0 & 2 & 6t & 12t^2 \end{bmatrix} \cdot p_s = \begin{bmatrix} s_0 \\ \dot{s}_0 \\ \ddot{s}_0 \\ \dot{s}_f \\ \ddot{s}_f \end{bmatrix}, \tag{9}$$

$$p_d = \begin{bmatrix} a_d^0 & a_d^1 & a_d^2 & a_d^3 & a_d^4 & a_d^5 \end{bmatrix}^T, \tag{10}$$

$$p_s = \begin{bmatrix} a_s^0 & a_s^1 & a_s^2 & a_s^3 & a_s^4 \end{bmatrix}^T, \tag{11}$$

where t is the time variable, p_d represents the coefficients of quintic polynomials for lateral planning, $\begin{bmatrix} d_0 & \dot{d}_0 & \ddot{d}_0 & d_{fn} & \dot{d}_{fn} & \ddot{d}_{fn} \end{bmatrix}^T$ is the boundary condition of a quintic polynomial, p_s represents the coefficients of quartic polynomials for longitudinal planning, and $\begin{bmatrix} s_0 & \dot{s}_0 & \ddot{s}_0 & \dot{s}_f & \ddot{s}_f \end{bmatrix}^T$ is the boundary condition of a quartic polynomial. In the Frenet coordinate system [31], s represents the arc length along the road, used for longitudinal position description, while the lateral position is represented by the offset d perpendicular to the path. The polynomial coefficients p_d and p_s can be solved by substituting the planning time T_c into the polynomial planning equations. Specifically, the parameters d_0, $\dot{d}_0$, and $\ddot{d}_0$ represent the initial position, initial velocity, and initial acceleration, respectively; d_{fn}, $\dot{d}_{fn}$, and $\ddot{d}_{fn}$ represent the final position, final velocity, and final acceleration, respectively. Similarly, the parameters s_0, $\dot{s}_0$, and $\ddot{s}_0$ represent the initial position, initial velocity, and initial acceleration, respectively; s_f, $\dot{s}_f$, and $\ddot{s}_f$ represent the final position, final velocity, and final acceleration, respectively. a_d^i ($i = 0, 1, 2, 3, 4, 5$) are the coefficients of the quintic polynomial, and a_s^i ($i = 0, 1, 2, 3, 4$) are the coefficients of the quartic polynomial. Using the initial and final boundary conditions in the above matrix equations, the polynomial coefficients can be solved to obtain the polynomial forms for trajectory planning. The quintic polynomial is used for lateral planning, and the quartic polynomial is used for longitudinal planning.

The design of the action space needs to consider the goal of the autonomous driving task. The action space of the RL problem is designed as follows:

$$\mathrm{A} = \begin{bmatrix} d_{fn} & \dot{s}_f \end{bmatrix}, \tag{12}$$

where d_{fn} and $\dot{s}_f$ are the expected lateral offset and expected longitudinal velocity after time T_c. A simple proportional–integral–derivative controller is applied for tracking the desired trajectory generated by the RL algorithm [32].

Reward design

The design of the reward function is very important for RL algorithms. The reward function is used to guide and evaluate the agent's behavior, which has a direct impact on the performance of the algorithm. In this paper, the reward function with adversarial nature is designed to approach the performance boundary of the ego vehicle by guiding the environment agent to continuously generate adversarial behaviors.

Firstly, the adversarial policy of the environment agent should improve the collision risk with the ego vehicle as much as possible. The artificial potential field method is a kind of virtual force method [33], the basic idea of which is to design the movement of the robot in the surrounding environment into an abstract artificial gravitational field. The artificial potential field contains 2 kinds of force fields: the attractive field formed by the position of the moving target and the repulsive field formed by the obstacle. Therefore, inspired by the improved artificial potential field theory [34], the collision risk of the ego vehicle is quantified.

The risk reward r_1 is designed as follows:

$$r_1 = -\sum_{i=1}^{m} \min\left(\exp\left(\frac{\left(\min\left(\max\left(\Delta d - L_a, r_{\min}\right), r_{\max}\right)\right)^2}{\delta_1^2} + \frac{\left(\min\left(\max\left(\Delta s - L_b, r_{\min}\right), r_{\max}\right)\right)^2}{\delta_2^2}\right) \cdot k_f \cdot g + b, 0\right) \tag{13}$$

where $r_{\min}$ and $r_{\max}$ are the safe distance penalty lower bound constant and the safe distance penalty upper bound constant, respectively. δ_1 and δ_2 are the lateral repulsion influence factor and the longitudinal repulsion influence factor, respectively. L_a is the vehicle width, L_b is the vehicle length, and k_f is the scale factor. In order to keep the reward value in a reasonable range, g is defined as the linear mapping proportion and b is defined

as the linear mapping deviation. g and b can be expressed as follows:

$$g = -\frac{\rho_n}{k_f \cdot S - k_f}, \tag{14}$$

$$b = \rho_n - g \cdot k_f, \tag{15}$$

where ρ_n is the safe distance penalty factor. S is the proportional correction factor, which can be expressed as

$$S = \exp\left(\frac{\left(d_{thre} - L_a\right)^2}{\delta_1^2} + \frac{\left(s_{thre} - L_b\right)^2}{\delta_2^2}\right), \tag{16}$$

where d_{thre} and s_{thre} are the lateral and longitudinal threshold values of the safe distance penalty, respectively.

The vehicle speed is encouraged to be maintained within a reasonable range. In the start-up stage, the reward function is designed to guide the vehicle to accelerate from 0, while in the speed maintenance stage, the reward function is designed to maintain in a preset speed interval. In order to design the adversarial reward to harm the longitudinal performance of the vehicle, the minimum speed reward r_2 is designed as follows:

$$r_2 = -\rho_L \times \kappa_L - \rho_H \times \kappa_H, \tag{17}$$

where ρ_L is the start-up stage reward factor, ρ_H is the speed maintenance stage reward factor, $\kappa_L = v_s / v_{\min}$ is the ratio between the longitudinal velocity v_s and the desired minimum velocity v_{min}, and $\kappa_H = \left(v_s - v_{\min}\right) / \left(v_{\max} - v_{\min}\right)$ is the ratio of the vehicle speed v_s to the desired maximum speed v_{max} based on the desired minimum speed v_{min}.

Causing the collision accident is considered to be the most serious consequence that can result from the adversarial behavior produced by the environment agent. Therefore, when the collision occurs, the collision reward r_3 is set as follows.

$$r_3 = \begin{cases} 0 & \text{not collision,} \\ \rho_{coll} & \text{collision,} \end{cases} \tag{18}$$

where ρ_{coll} is the collision reward factor. ρ_{coll} is set to a positive value to indicate that policies that receive this reward are encouraged.

The total reward function is

$$R = -r_1 + r_2 + r_3 \tag{19}$$

All reward function parameters are listed in Table 2.

Table 2. Reward function parameters setting

Reward function parameter terms	Symbol and value
Safe distance penalty lower bound constant	$r_{\min} = 0.0$
Safe distance penalty upper bound constant	$r_{\max} = 150.0$
Lateral repulsion influence factor	$\delta_1 = 8.0$
Longitudinal repulsion influence factor	$\delta_2 = 10.0$
Scale factor	$k_f = 0.001$
Vehicle width	$L_a = 2.077$ m
Vehicle length	$L_b = 5.037$ m
Safe distance penalty factor	$\rho_n = -18.0$
Lateral threshold value of the safe distance penalty	$d_{thre} = 0.8$
Longitudinal threshold value of the safe distance penalty	$S_{thre} = 20.0$
Start-up stage reward factor	$\rho_L = 0.5$
Speed maintenance stage reward factor	$\rho_H = 4.0$
Expected minimum speed	$v_{\min} = 7.5$ m/s
Expected maximum speed	$v_{\max} = 22.0$ m/s
Collision reward factor	$\rho_{coll} = 200.0$

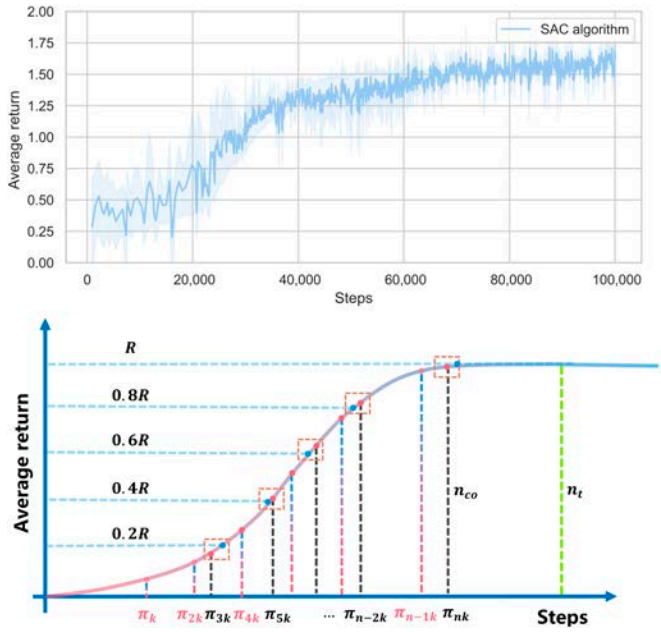

Fig. 8. Construction of the policy group. SAC, soft actor critic.

Data generation of varied difficulty scenarios

In the "Environment agent policy search" section, we introduce the concept of environment agent to realize the adversarial policy search by combining logic rules with RL. However, due to the black-box nature of data-driven methods, while adversarial actions can be generated, the difficulty of generating adversarial actions is difficult to quantify accurately, which limits the rationality of adversarial scenario generation.

In this section, a data generation method based on scenarios of varying difficulty is presented. The method uses the performance of different stages in the policy search convergence process as a reference to quantify the adversarial intensity, thereby achieving a quantitative representation of scenario difficulty. The model parameters of the environment agent trained on different stages are updated and saved and then output to the constructed policy group. The policy group is used to generate data that forms the basis for training the scenario difficulty quantitative representation model.

Policy group construction

As shown in Fig. 8, the schematic of the average return during the policy search of the environment agent is presented. The horizontal axis shows the number of training steps, and the vertical axis shows the average return. The training details can be seen in Results and Discussion.

An RL training process with stable convergence can be divided into 2 phases, i.e., the performance improvement phase and the convergence stabilization phase. In the performance improvement phase, the average return is still continuously increasing, which indicates that the policy search is still ongoing and the model parameters are still being updated to produce a better performance. In the convergence stabilization phase, however, the average return remains basically unchanged, indicating that the policy search is basically over, and the obtained policy is already the optimal policy that the current algorithm can achieve.

Considering that the intensity of the environment agent's adversarial behavior against the ego vehicle may increase nonlinearly with the training process, a policy filtering method is proposed to ensure that the policies within the policy group can generate distinguishable and reasonable adversarial scenarios with different intensities. The policy filtering algorithm is shown in Algorithm 1 and Fig. 8.

The initialization and parameter setting of the algorithm are performed in lines 1 to 3. In lines 4 and 5, through trial training of the environment agent, reasonable total steps n_t, performance improvement steps n_{co}, and maximum average return R are obtained.

The policy search of the environment agent π_E is shown in line 7, and the parameter θ of π_E is updated using Eq. 4. Lines 8 to 12 represent the sampling operations conducted on model parameters during the convergence process of the algorithm. The sampled model parameters are placed in set Ξ. The sampling interval is k; i.e., the model is sampled every k rounds. The sampled model π_k and the corresponding training steps e_k form a binary group and are placed in set Ξ, which can be denoted as

$$\Xi = \left\{\Pi_k, \Pi_{2k}, \Pi_{3k}, \ldots, \Pi_{nk}\right\}, \tag{20}$$

where $\Pi_k = \left\{\pi_k, e_k\right\}$, $n = len(\Xi)$.

As shown in Fig. 8, the bottom of the horizontal axis represents the sampling models $\pi_k, \pi_{2k}, \ldots, \pi_{nk}$ at different phases in set Ξ, corresponding to the positions at different time steps $e_k, e_{2k}, \ldots, e_{nk}$.

Algorithm 1 Policy Group Construction

1: **Input:** Ego vehicle policy π_A, scenario dynamics S_c;
2: **Output:** Total steps n_t, performance enhancement steps n_{co}, maximum average return R
3: Initialize Policy group $O = \emptyset$, set $\Xi = \emptyset$, environment agent network π_E with weights θ, episode number $\vartheta = 0$;
4: Set parameters Sampling interval k;
5: Execute Trial train environment agent π_E under π_A and S_c
6: **for** *i=1, 2, ...,*n_t **do**
7: **Execute** policy search, update environment agent network π_E using Eq. 4
8: **if** $i < n_{co}$ **and** mod $(\vartheta, k) = 0$ **then**
9: $e_k = i$
10: Collect $\Pi_k = (\pi_k, e_k)$
11: $\Xi \leftarrow \Xi \cup \{\Pi_k = (\pi_k, e_k)\}$
12: **if** episode end **then**
13: $\vartheta = \vartheta + 1$
14: **for** *i=1, 2, 3, 4, 5* **do**
15: $n_l^i = \Gamma^{-1}(0.2i \cdot R)$
16: **for** *j=1, 2, ...,len*(Ξ) **do**
17: $\tau = \arg\min\left(n_l^i - \Xi[j][2]\right)$
18: $O \leftarrow O \cup \Xi[\tau]$

In lines 13 to 17, policy filtering is performed to ensure that the intensity of the adversarial behaviors produced by the environment agent models in policy group O grows linearly. To quantify the intensity of adversarial strength, the maximum average return R is used. Figure 8 illustrates the policy filtering process. First, the mapping relation between the average return curve and the training steps is defined as

$$R_s = \Gamma\left(n_s\right), \tag{21}$$

where R_s is the corresponding average return value at n_s steps.

In lines 13 and 14, the maximum average return R is linearly split, as shown in Fig. 8, to obtain the number of training steps corresponding to the dividing line (the horizontal coordinates corresponding to the blue circle center point). In lines 15 and 16, a find algorithm is executed to find the index τ of the sampled model that is closest to n_l^i in $e_k, e_{2k}, \ldots, e_{nk}$ (i.e., closest pink center point to the blue center point in Fig. 8).

In line 17, the sampling models are extracted from set Ξ according to index τ and deposited into O to finalize the construction of the policy group. The sampled models that have been filtered and deposited into policy group O are shown as black dashed lines in Fig. 8.

Varied difficulty scenario dataset

The constructed policy group O is used to build a varied difficulty scenario dataset, and this process is carried out in a simulation environment. The parameters of the simulation environment are randomly configured, which is to simulate more diverse real-world inputs and ensure the generalization of subsequent model training.

The pseudocode of the construction for the varied difficulty scenario dataset is shown in Algorithm 2.

Algorithm 2 Varied Difficulty Scenario Dataset Construction

1: **Input:** Policy group O, traffic model τ_{tf} with parameters σ;
2: Initialize simulation environment S_{world}, varied difficulty scenario dataset Ψ;
3: Set parameters same category data size n_{sc}, episode number $n_{episode}$;
4: **for** *i=1, 2, ...,len*(O) **do**
5: **for** *j=1, 2, ...,*$n_{episode}$ **do**
6: **Deploy** the policy $O[i]$ to the environment agent
7: **Random choose** traffic model parameters σ
8: **Generate** random traffic flow using τ_{tf} in simulation
9: Using environment agent to collect $(S, 0.2 * i, \mathrm{a})$
10: $\Psi \leftarrow \Psi \cup (S, 0.2 * i, \mathrm{a})$
11: **if** steps num $> i * n_{sc}$ **then**
12: break

First, line 1 initializes the simulation environment S_{world} and varied difficulty scenario dataset $\boldsymbol{\Psi}$. Lines 2 and 3 determine the input and parameter setting of the algorithm. Lines 4 to 12 represent a complete dataset construction process. In line 6, the policies in the policy group O are deployed to the environment agent. In lines 7 and 8, the random traffic flow τ_{tf} is generated in the simulation environment by randomizing the parameters of the traffic model σ. In lines 9 and 10, the environment agent is used to collect the ternary $(S, 0.2 * i, \mathrm{a})$ into $\boldsymbol{\Psi}$, where $0.2 * i$ denotes the difficulty factor of the scenario. Lines 11 and 12 indicate that when the collected data size of the same category exceeds n_{sc}, the data collection for the corresponding difficulty level is finished. The algorithm then jumps out the current inner loop, proceeds to deploy the environment agent's policy for the next difficulty level, and continues until varied difficulty scenario datasets $\boldsymbol{\Psi}$ are completely collected.

Quantitative representation of scenario difficulty

To obtain an environment agent model with an adversarial behavior, a quantitative representation model is constructed. The model is based on a deep neural network with a transformer decoding architecture. As shown in Fig. 5, the input of the model is the state s and the scenario difficulty factor X_{scd}, and the output is the action a of the environment agent.

In recent years, the transformer model has gained much attention [35,36], and its core idea is the self-attention mechanism. The key to the self-attention mechanism is to assign different weights to certain positions, thus enabling the model to better capture long-distance dependencies and global information in the input sequence. This mechanism not only contributes to a deeper understanding of the intrinsic structure of the sequence but also provides interpretability for the model, allowing it to focus on important parts of the sequence.

The proposed quantitative mode is constructed based on the transformer encoder model [37]. The transformer encoder consists of alternating layers of self-attention and multilayer perceptron block. Layer norm (LN) is applied before every block, and residual connections after every block.

The input of the neural network includes the scenario difficulty factor X_{scd} and the relative information between the environment agent and the ego vehicle. The input vector can be expressed as follows:

$$\mathrm{X} = \left[X_{scd} s\right] = \left[X_{scd}, \Delta s, \Delta d, \Delta v_s, v_s a_s\right] \tag{22}$$

The standard transformer model takes a 1-dimensional sequence of token embeddings as input. The embedding projection is then applied to the extracted features from the input vector.

$$h_0 = \left[X_{scd} E; \Delta s E; \Delta d E, \Delta v_s E, v_s E, a_s E\right], \quad E \in {}^{P \times D} \tag{23}$$

The transformer encoder model can be expressed as

$$\mathbf{z}'_{\ell} = \mathrm{MSA}\left(\mathrm{LN}\left(\mathbf{z}_{\ell-1}\right)\right) + \mathbf{z}_{\ell-1} \quad \ell = 1 \ldots L \tag{24}$$

$$\mathbf{y} = \mathrm{LN}\left(\mathbf{z}_L^0\right) \tag{25}$$

In Eq. 24, the self-attention criterion divides the input embedding into 3 vectors V, K, and Q [38]. The scaled dot-product attention is calculated according to Eq. 26.

$$\Theta = Attention(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{26}$$

where Θ is the score matrix, Q is a query vector, K is a key vector, V is a value vector, and d_k is a normalization.

The training process of scenario difficulty quantitative representation model φ is shown in Algorithm 3. φ is trained with data from varied difficulty scenario dataset Ψ. In line 5, a batch B is randomly sampled from φ. In lines 7 and 8, the loss function is calculated by the mean square error between the predicted and true values, where a and $\hat{a}$ denote the ground truth and predicted values, respectively. The model is trained using the Adams optimizer.

The score matrix of the transformer can be used to analyze the focus of the model when processing sequences, thus improving the interpretability of the model's behavior. The score matrix Θ can calculated using Eq. 26.

Algorithm 3 Training process of scenario difficulty quantitative representation model

1: **Input:** Varied difficulty scenario dataset Ψ;
2: Initialize quantitative representation model φ with weights θ_{sd};
3: Set parameters batch size n_b, epoch number n_{epoch};
4: **for** i=1, 2, …,n_{epoch} **do**
5: **Random sample** $B = (X_{scd}, s)_{i=0:n_b-1}$ from Ψ
6: Predict $\hat{a} = \varphi_{\theta_{sd}}(X_{scd}, s)$
7: Compute MSE loss $\mathcal{L} = \|a - \hat{a}\|^2$
8: Update θ_{sd} by $\hat{\nabla}_{\theta_{sd}} \mathcal{L}_{\varphi}(\theta_{sd})$ using Adams optimizer
9: **end for**

$$\Theta = \begin{bmatrix} \mu_{scd}^{1\times1} & \mu_{S_{[1]}}^{1\times2} & \cdots & \mu_{S_{[n]}}^{1\times(n+1)} \\ \mu_{S_{[1]}}^{2\times1} & \mu_{a}^{2\times2} & \cdots & \mu_{a}^{2\times(n+1)} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{S_{[n]}}^{n+1\times1} & \mu_{a}^{n+1\times2} & \cdots & \mu_{a}^{n+1\times(n+1)} \end{bmatrix}, \tag{27}$$

where μ is a scalar and the bottom-right symbol is used to distinguish the source of μ. Θ can be used to characterize the feature correlation between each state. For the scenario difficulty quantitative representation model, the state correlation between the scenario difficulty factor X_{scd} and the relative information between the environment agent and the ego vehicle is the most noteworthy. By summing the elements along columns in matrix Θ, a new matrix $\tilde{\Theta}^{1\times(n+1)}$ can be obtained.

In order to obtain an ordering of the characteristic correlation between the scenario difficulty factor X_{scd} and the other states s, $\tilde{\Theta}^{1\times(n+1)}$ is sorted and the indexes of the array elements sorted are obtained, as shown in Eq. 28:

$$\Lambda = argsort\left(\tilde{\Theta}^{(n+1)\times(n+1)}\right), \tag{28}$$

where the index matrix Λ is the vector of $1 \times n$ and the corresponding sorted feature contribution value is $\Upsilon = \tilde{\Theta}^{1\times(n+1)}[\Lambda]$.

Acknowledgments

Funding: We would like to thank the National Key R&D Program of China under Grant No. 2022YFB2502900, the National Natural Science Foundation of China (Grant Number: U23B2061), the Fundamental Research Funds for the Central Universities of China, and the Xiaomi Young Talent Program, and we thank the reviewers for the valuable suggestions.

Author contributions: S.Y. and Y.H. conceived the idea and designed the experiments. S.Y. and C.W. wrote the original draft. Y.Z., Y.Y., S.E.L., and H.C. reviewed and revised the draft. H.C. supervised and led the planning and execution of research activities.

Competing interests: The authors declare that they have no competing interests.

Data Availability

The data that support the findings of this study are available from the School of Automotive Studies, Tongji University. Restrictions apply to the availability of these data, which were used under license for this study. Data access inquiries can be sent to shuo_yang@tongji.edu.cn.

References

1. Feng S, Sun H, Yan X, Zhu H, Zou Z, Shen S, Liu HX. Dense reinforcement learning for safety validation of autonomous vehicles. *Nature*. 2023;615(7953):620–627.
2. He X, Lv C. Towards safe autonomous driving: Decision making with observation-robust reinforcement learning. *Automot Innov*. 2023;6:509–520.
3. Huang Y, Yang S, Wang L, Yuan K, Zheng H, Chen H. An efficient self-evolution method of autonomous driving for any given algorithm. *IEEE Trans Intell Transp Syst*. 2023;25(1):602–612.
4. Fan C, Yao L, Zhang J, Zhen Z, Wu X. Advanced reinforcement learning and its connections with brain neuroscience. *Research*. 2023;6:Article 0064.
5. Cao Z, Jiang K, Zhou W, Xu S, Peng H, Yang D. Continuous improvement of self driving cars using dynamic confidence-aware reinforcement learning. *Nat Mach Intell*. 2023;5:145–158.
6. Yang S, Huang Y, Li L, Feng S, Na X, Chen H, Khajepour A. How to guarantee driving safety for autonomous vehicles in a real-world environment: A perspective on self-evolution mechanisms. *IEEE Intell Trans Syst Mag*. 2024;16(2):41–54.
7. Li X, Wang Z, Huang Y, Chen H. A survey on self-evolving autonomous driving: A perspective on data closed-loop technology. *IEEE Trans Intell Veh*. 2023;8(11):4613–4631.
8. Riedmaier S, Ponn T, Ludwig D, Schick B, Diermeyer F. Survey on scenario-based safety assessment of automated vehicles. *IEEE Access*. 2020;8:87456–87477.
9. Kalra N, Paddock SM. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transp Res A Policy Pract*. 2016;94:182–193.
10. Ding W, Xu C, Arief M, Lin H, Li B, Zhao D. A survey on safety-critical driving scenario generation—A methodological perspective. *IEEE Trans Intell Transp Syst*. 2023;24(7):6971–6988.
11. Wu J, Xing X, Xiong L, Chen J. Accelerated testing and evaluation of autonomous vehicles based on dual surrogates. *Automot Innov*. 2024;7:390–402.
12. Rocklage E, Kraft H, Karatas A, Seewig J. Automated scenario generation for regression testing of autonomous vehicles. In: *2017 IEEE 20th international conference on intelligent transportation systems (ITSC)*. New York (NY): IEEE; 2017. p. 476–483.
13. de Gelder E, Hof J, Cator E, Paardekooper J-P, den Camp OO, Ploeg J, de Schutter B. Scenario parameter generation method and scenario representativeness metric for scenario-based assessment of automated vehicles. *IEEE Trans Intell Transp Syst*. 2022;23(10):18794–18807.
14. Feng S, Yan X, Sun H, Feng Y, Liu HX. Intelligent driving intelligence test for autonomous vehicles with naturalistic and adversarial environment. *Nat Commun*. 2021;12(1):Article 748.
15. Luo YT, Li PQ, Li DT, Peng YG, Geng ZG, Xie SH, Li Y, Alù A, Zhu J, Zhu XF. Probability-density-based deep learning paradigm for the fuzzy design of functional metastructures. *Research*. 2020;2020:Article 8757403.
16. Wheeler TA, Kochenderfer MJ, Robbel P. Initial scene configurations for highway traffic propagation. In: *2015 IEEE 18th international conference on intelligent transportation systems*. New York (NY): IEEE; 2015. p. 279–284.
17. Demetriou A, Alfsvag H, Rahrovani S, Chehreghani MH. A deep learning framework for generation and analysis of driving scenario trajectories. *SN Comput Sci*. 2023;4(3):251.
18. Feng L, Li Q, Peng Z, Tan S, Zhou B. TrafficGen: Learning to generate diverse and realistic traffic scenarios. In: *2023 IEEE international conference on robotics and automation (ICRA)*. New York (NY): IEEE. 2023, p. 3567–3575.
19. Wang J, Pun A, Tu J, Manivasagam S, Sadat A, Casas S, Ren M, Urtasun R. AdvSim: Generating safety-critical scenarios for self-driving vehicles. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. New York (NY): IEEE; 2021. p. 9909–9918.
20. Hanselmann N, Renz K, Chitta K, Bhattacharyya A, Geiger A. KING: Generating safety critical driving scenarios for robust imitation via kinematics gradients. In: Avidan S, Brostow G, Cissé M, Farinella GM, Hassner T, editors. *Computer vision—ECCV 2022: 17th European conference, Tel Aviv, Israel, October 23–27, 2022, proceedings, part XXXVIII*. Cham (Switzerland): Springer; 2022. p. 335–352.
21. Ding W, Chen B, Xu M, Zhao D. Learning to collide: An adaptive safety-critical scenarios generating method. In: *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. New York (NY): IEEE; 2020. p. 2243–2250.
22. Zhang M, Zhang Y, Zhang L, Liu C, Khurshid S. DeepRoad: GAN-based metamorphic testing and input validation framework for autonomous driving systems. In: *Proceedings of the 33rd ACM/IEEE international conference on automated software engineering*. New York (NY): IEEE; 2018. p. 132–142.
23. Jia L, Yang D, Ren Y, Qian C, Feng Q, Sun B, Wang Z. A dynamic test scenario generation method for autonomous vehicles based on conditional generative adversarial imitation learning. *Accid Anal Prev*. 2024;194:Article 107279.
24. Rempe D, Philion J, Guibas LJ, Fidler S, Litany O. Generating useful accident-prone driving scenarios via a learned traffic prior. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. New York (NY): IEEE; 2022. p. 17305–17315.
25. Hao K, Cui W, Luo Y, Xie L, Bai Y, Yang J, Yan S, Pan Y, Yang Z. Adversarial safety-critical scenario generation using naturalistic human driving priors. *IEEE Trans Intell Veh*. 2023;1–6.
26. Dosovitskiy A, Ros G, Codevilla F, Lopez A, Koltun V. CARLA: An open urban driving simulator. In: Levine S, Vanhoucke V, Goldberg K, editors. *Conference on robot learning*. PMLR; 2017. p. 1–16.
27. Van der Maaten L, Hinton G. Visualizing data using t-SNE. *J Mach Learn Res*. 2008;(86):9, 2579–2605.
28. Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: *International conference on machine learning*. PMLR; 2018, p. 1861–1870.
29. Haarnoja T, Zhou A, Hartikainen K, Tucker G, Ha S, Tang J, Kumar V, Zhu H, Gupta A, Abbeel P, et al. Soft actor-critic algorithms and applications. arXiv. 2018. https://arxiv.org/abs/1812.05905
30. Wu CS, Chiu ZY, Liu JS. Time-optimal trajectory planning along parametric polynomial lane-change curves with bounded velocity and acceleration: Simulations for a unicycle based on numerical integration. *Model Simul Eng*. 2018;2018:Article 9348907.
31. Werling M, Ziegler J, Kammel S, Thrun S. Optimal trajectory generation for dynamic street scenarios in a Frenet frame. In: *2010 IEEE international conference on robotics and automation*. New York (NY): IEEE; 2010. p. 987–993.
32. Farag W. Complex trajectory tracking using PID control for autonomous driving. *Int J Intell Trans Syst Res*. 2020;18: 356–366.

33. Khatib O. Real-time obstacle avoidance for manipulators and mobile robots. *Int J Robot Res*. 1986;5:90–98.
34. Shi P, Zhao Y. Global path planning for mobile robot based on improved artificial potential function. In: *2009 IEEE international conference on automation and logistics*. New York (NY): IEEE; 2009. p. 1900–1904.
35. Hu Y, Yang J, Chen L, Li K, Sima C, Zhu X, Chai S, Du S, Lin T, Wang W, et al. Planning-oriented autonomous driving. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. New Yorl (NY): IEEE; 2023. p. 17853–17862.
36. Vaswani A, Shazeer N, Parmar N, Uskoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I. Attention is all you need. *Adv Neural Info Process Syst*. 2017;30:6000–6010.
37. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, et al. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv. 2020. https://arxiv.org/abs/2010.11929
38. Li G, Qiu Y, Yang Y, Li Z, Li S, Chu W, Green P, Li SE. Lane change strategies for autonomous vehicles: A deep reinforcement learning approach based on transformer. *IEEE Trans Intell Veh*. 2022;8(3):2197–2211.