Complex & Intelligent Systems (2024) 10:1715–1732
https://doi.org/10.1007/s40747-023-01238-6

ORIGINAL ARTICLE

Actor-critic objective penalty function method: an adaptive strategy for trajectory tracking in autonomous driving

Bo Wang[1] · Fusheng Bai[1] · Ke Zhang[1]

Received: 19 March 2023 / Accepted: 3 September 2023 / Published online: 30 September 2023

Abstract
Trajectory tracking is a key technology for controlling the autonomous vehicles effectively and stably to track the reference trajectory. How to handle the various constraints in trajectory tracking is very challenging. The recently proposed generalized exterior point method (GEP) shows high computational efficiency and closed-loop performance in solving the constrained trajectory tracking problem. However, the neural networks used in the GEP may suffer from the ill-conditioning issue during model training, which result in a slow or even non-converging training convergence process and the control output of the policy network being suboptimal or even severely constraint-violating. To effectively deal with the large-scale nonlinear state-wise constraints and avoid the ill-conditioning issue, we propose a model-based reinforcement learning (RL) method called the actor-critic objective penalty function method (ACOPFM) for trajectory tracking in autonomous driving. We adopt an integrated decision and control (IDC)-based planning and control scheme to transform the trajectory tracking problem into MPC-based nonlinear programming problems and embed the objective penalty function method into an actor-critic solution framework. The nonlinear programming problem is transformed into an unconstrained optimization problem and employed as a loss function for model updating of the policy network, and the ill-conditioning issue is avoided by alternately performing gradient descent and adaptively adjusting the penalty parameter. The convergence of ACOPFM is proved. The simulation results demonstrate that the ACOPFM converges to the optimal control strategy fast and steadily, and perform well under the multi-lane test scenario.

Keywords Autonomous driving · Trajectory tracking · Model predictive control (MPC) · Reinforcement learning (RL) · Objective penalty function method

Introduction

Autonomous driving has attracted increasingly more attention from practitioners and researchers [1]. The primary research areas for autonomous driving include three major aspects: visual perception, planning and decision, and motion control [2]. Trajectory tracking is a key technology for precisely controlling autonomous vehicles. As a fundamental part of the motion control module, trajectory tracking aims to calculate the vehicle control commands based on the reference trajectory obtained by the path planning module to achieve accurate tracking of the reference trajectory. The performance of trajectory tracking directly determines, to a large extent, the performance of autonomous vehicles, which involves driving safety, passenger comfort, travel efficiency, energy consumption and so on. The trajectory tracking problem in autonomous driving is very challenging as the system involved is typically high nonlinear and contains large-scale state-wise constraints [3]. How to mathematically model the trajectory tracking problem properly, set up the obstacle avoidance strategy, reduce the huge computational burden in the solution process, and minimize the trajectory tracking error are crucial issues in the trajectory tracking research [4, 5].

The current vehicle trajectory tracking control methods mainly include sliding mode control (SMC), proportion integration differentiation (PID) control, and model predictive

✉ Fusheng Bai
fsbai@cqnu.edu.cn

Bo Wang
wb1999430@163.com

Ke Zhang
20208023@cqnuedu.cn

1 National Center for Applied Mathematics in Chongqing, Chongqing Normal University, Chongqing 401331, China

control (MPC) [6–8], etc. SMC is a nonlinear variable structure control, in which the composition of the system is constantly changing with the system state, so that the system always changes according to the employed slip mode state. SMC has the advantages of fast response, insensitivity to parameter changes and perturbations, and no need for online system identification [9], but it is prone to chatter, which is the most noticeable issue affecting the practical application of SMC [10]. The chattering issue can be lessened by approximating the discontinuous terms via continuous functions like saturation function, arctan function, and hyperbolic tangent function [11–13]. Although some improvements obtained using these modified SMC methods, there are still some problems associated with the presence of external disturbances [14–16]. PID control is used broadly in industry because it is simple and does not require a mathematical model [17], but it is not applicable to nonlinear systems and the tuning of parameters is complicated, which makes it hard to guarantee the accuracy and robustness of the control [18]. MPC employs the nonlinear dynamics model of the control system and predicts the output behavior of the system in the future period by solving the optimal control problem with constraints [19], however its tracking performance is sensitive to the accuracy of the prediction model, and the nonlinear MPC has high demand for computational resources [20].

Reinforcement learning (RL) solves sequential decision-making problems via a trial-and-error process interacting with the environment [21]. Model-based RL builds environment models in which trial-and-error can take place without real costs and can improve RL algorithms significantly by making them more sample-efficient and thus reducing errors [22]. Deep neural networks (DNN) have been commonly used as function approximators in RL, which mitigate against the state-action space explosion as the number of state-action pairs recorded grows [23], and generalize prior experience to previously unseen state-action pairs, leading to the birth of deep reinforcement learning (DRL). Actor-critic methods for RL are hybrid methods that combine the advantages of policy-based and value-based algorithms, where the ‘actor’ is the term for the policy structure in charge of choosing actions and the ‘critic’ is the estimated value function to evaluates the actor’s actions. Zhao et al. [24] proposed a model-based actor-critic framework for optimal tracking control of robotic systems, which overcomes the issue of long learning cycles for control strategies. The actor-critic based solution framework is widely used in autonomous driving tasks, especially to solve the problem for which the training process of RL or DRL can be hard when the state space is enormous [25]. Recently, given the uncertainties in the driving conditions, a soft actor-critic scheme was used in [26] to solve the decision-making and planning problem with continuous action space. Note that in general the DRL lacks closed-loop stability analysis, state constraints satisfiability, and significant weight initialization capacity [27]. In order to deal with these issues, the prospect of using MPC-based RL has been proposed and justified in [28], where using MPC as the function approximation for the optimal policy in RL is suggested. MPC based strategies can effectively deal with the multivariable constraints in autonomous vehicle tracking under uncertain environments by building systems that satisfy state constraints and safety requirements [29].

Recently, Guan et al. [30] proposed an integrated decision and control (IDC) framework for autonomous vehicles, which has better interpretability and computational efficiency compared with the traditional decomposed scheme [31] and the end-to-end scheme [32]. In addition, it is applicable in different driving scenarios and tasks. A model-based actor-critic RL algorithm, called the generalized exterior point method (GEP) is presented in [30] for solving the formulated MPC-based optimal control problem with large-scale state-wise constraints in a rolling optimization process to minimize the trajectory tracking error. The GEP is an extension of the exterior point method in the optimization domain to the field of neural network (NN), which transforms the constrained optimal control problem into an unconstrained problem with penalties for safety violations and an actor-critic solution framework is employed to solve the problem. The purpose of the actor is to obtain an excellent policy NN, which aims to make trajectory tracking performance as good as possible without collision. The objective function with penalties is mapped as the parameterized loss function of the policy NN for model training, and the input is the state vector of the autonomous vehicle and the output is the control vector, thus giving the corresponding control commands for the autonomous vehicle. Meanwhile, the critic, with the function of judging state goodness, can be served as the path selector. Unlike the exterior point method in the optimization domain, the GEP relaxes the conditions to establish convergence of the exterior point method, eliminates the need to find global minima in each gradient descent step, and updates the NN parameters effectively under the guidance of the model. An approximate feasible optimal control strategy is obtained by implementing gradient descent and updating the penalty parameter alternatively. For the exterior point method, it can be shown that under certain conditions, when the penalty parameter increases to infinity, the optimal solution to the penalty problem is the optimal solution to the original problem [33]. However, as the penalty parameter increases, it would face the ill-conditioning issue [34, 35], including severe numerical instabilities [36], so the penalty parameter cannot be infinitely large. A similar issue arises for DNN training [37–39], where each iterative update of the network parameters is essentially a step of gradient descent. On the one hand, we do not know exactly how large the penalty parameters need to be. Theoretically speaking, the penalty parameters have to be increased to infinity to ensure convergence, but due to the actual computational limitations, the penalty parameters cannot be too large or too small [40]. On the other hand, as the penalty parameters keep increasing, the condition number of the Hessian matrix of the loss function becomes larger and larger, the correlation between the columns of the matrix becomes too high [41, 42], and the ill-conditioning issue in the training of the DNN becomes more and more serious, which results in a very slow or even non-converging training process [38, 39]. In other words, the control of DNN output may be suboptimal or even constraint-violating, so that we can hardly ensure the quality of the trained model.

In this paper, we propose a model-based RL algorithm called the actor-critic objective penalty function method (ACOPFM) to solve the trajectory tracking problem. The ACOPFM can effectively deal with the large-scale nonlinear state-wise constraints and avoid the ill-conditioning issue caused by the increasing condition number of the Hessian matrix during the model training of the policy network of the GEP. The ACOPFM transforms the nonlinear programming problem into a parametric, unconstrained optimization problem and employs the objective penalty function as a loss function for policy network updates to find the optimal control strategy by alternately performing gradient descent and adaptively adjusting the penalty parameter. The convergence of ACOPFM is proved. Meanwhile, the ACOPFM can improve the computational efficiency of the solution process and update the NNs efficiently under the guidance of the model. To the best of our knowledge, this is the first time that the objective penalty function method is extended to the RL framework. Moreover, we conduct extensive simulation experiments via the joint platform consisting of CARLA and SUMO. During the offline training, the proposed ACOPFM converges faster and more steadily to the optimal control strategy with the GEP as the baseline. During the online testing, our trained autonomous vehicle can successfully complete the trajectory tracking task in the overtaking challenge under the multi-lane test scenario with frequent vehicle interactions.

The remainder of this paper is organized as follows. Section "Related work" briefly introduces the penalty function methods in the optimization field, and the development of the integration of MPC and RL. Section "Planning and control scheme" describes the planning and control scheme we have adopted based on the IDC framework. Section "Design of the nonlinear programming problem" presents the modeling of the trajectory tracking problem into the MPC-based nonlinear programming problem. Section "ACOPFM framework" presents the ACOPFM solution for the modeled trajectory tracking problem under the RL framework, and its theoretical convergence analysis. Section "Simulation experiment" reports the numerical results of the offline training and online testing of the NN trained using the ACOPFM for the multi-lane simulation scene. Finally, the conclusions of the present work and some suggestions for the future work are given in Sect. "Conclusion and future work".

Related work

In this section, we introduce firstly the penalty function methods, including the development of the objective penalty function methods in recent decades, then the integration of MPC and RL, including its applications in the field of autonomous driving.

Penalty function methods

Consider the following constrained optimization problem:

$$\begin{aligned} \min \quad & f(x), \\ \text{s.t.} \quad & g_c(x) \le 0, \quad c \in \boldsymbol{C}, \end{aligned} \tag{P}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function, $g_c : \mathbb{R}^n \to \mathbb{R}$, $c \in \boldsymbol{C}$ are constraints. The set

$$X = \left\{ x \in \mathbb{R}^n \,\middle|\, g_c(x) \le 0, c \in \boldsymbol{C} \right\}$$

is called the feasible set of (P).

The problem finds applications in various fields such as artificial intelligence and transportation [43, 44].

Penalty function methods are popular choices for solving problem (P). Based on the principle of converting a constrained optimization problem into a series of unconstrained optimization problems, penalty function methods are relatively simple and effective in finding the optimal solution [45, 46]. A commonly used penalty function for (P) is the l_2 penalty function, which is defined as:

$$F(x, \rho) = f(x) + \rho \sum_{c \in \boldsymbol{C}} \max\{g_c(x), 0\}^2,$$

where $\rho > 0$ is the penalty parameter. The corresponding penalty problem for (P) is given as follows:

$$\min F(x, \rho) \qquad \text{s.t.} \quad x \in \boldsymbol{R}^n. \tag{P$'$}$$

It can be shown that under certain conditions, when the penalty parameter increases to infinity, the optimal solution of the problem (P′) is also the optimal solution of the original problem (P) [46]. Therefore, when the penalty parameter keeps increasing, only a series of penalty problems need to be solved to obtain an approximate optimal solution of the original problem (P). However, the Hessian matrix of the penalty function will become increasingly ill-conditioning as the penalty parameter increases [34, 35, 47], which leads to numerical difficulties. Another commonly used penalty function is the l_1 penalty function, which is defined as:

$$F_1(x,\rho) = f(x) + \rho \sum_{c \in \boldsymbol{C}} \max\{g_c(x), 0\}.$$

The corresponding penalty problem for (P) is given as follows:

$$\min F_1(x,\rho) \qquad \text{s.t.} \quad x \in \boldsymbol{R}^n. \tag{P''}$$

Note that the l_1 penalty function is an exact penalty function (EPF) [48–51], which means that under some conditions, there is some ρ^* such that an optimal solution to (P) is also an optimal solution to (P′′) for all $\rho \geq \rho^*$. However, as the ρ^* is unspecified, it is necessary to gradually increase the penalty parameter to find the optimal solution to (P), and the ill-conditioning issue still occurs. Besides, the l_1 penalty function is nondifferentiable at the point x with some $g_c(x) = 0$. As the most powerful optimization methods require a differentiable cost function, this entails restrictions on choices of optimization methods to solve the exact penalty problem.

The idea of the objective penalty function method was first proposed in [52]. By adjusting the objective penalty parameter adaptively, the ill-conditioning issue can be effectively overcome.

The following objective penalty function was presented in [53]:

$$F(x,M) = (f(x) - M)^2 + \sum_{c \in \boldsymbol{C}} \max\{g_c(x), 0\}^p,$$

where $p > 1$, and an objective penalty function method (OPFM) was developed to solve (P) and the convergence of the corresponding algorithm was established under the assumption that X is connected and compact. In [54] a general objective penalty functions was proposed as follows:

$$F(x,M) = Q(f(x) - M) + \sum_{c \in \boldsymbol{C}} P(g_c(x)),$$

where $Q(t)$ is strictly decreasing on $[0, +\infty)$, $Q(t) > 0$ for $t \neq 0$, $Q(0) = 0$, and $P(t) > 0$ for $t > 0$, $P(t) = 0$ for $t \leq 0$. For example, $Q(t) = t^2$ and $P(t) = \max\{t, 0\}^2$. In [55] the modified $Q(t)$ and $P(t)$ were used to construct the objective penalty function, where $Q(t) = 0$ for $t < 0$ and $P(t)$ is further required to be strictly increasing on $[0, +\infty)$. The convergence results for the objective penalty function method are established in [55] without the compactness and connectivity assumption on the feasible set.

Integration of MPC and RL

MPC is a well-established control strategy that uses a mathematical model to predict the system behavior over a finite time horizon and optimize a performance criterion subject to constraints [8], but the performance of MPC is highly dependent on the accuracy of the model used for predictions. RL, on the other hand, is a machine learning technique that enables an agent to learn from its interactions with an environment and make decisions that maximize a reward signal [21], but many critical aspects still need to be tackled, including safety and stability issues [27].

The integration of MPC and RL can leverage the strengths of both approaches. MPC provides a framework for handling constraints and modeling complex systems, while RL allows for adaptive control in uncertain environments where traditional model-based methods may fail. Aswani et al. [57] first presented learning-based model predictive control (LBMPC) scheme, and used a linear model with bounds on its uncertainty to construct invariant sets that provided deterministic guarantees on robustness and safety. In [58], Koller et al. presented a learning-based model predictive control scheme that can provide provable high-probability safety guarantees. Gros and Zanon [28] proposed that a Nonlinear Model Predictive Control (NMPC) can be tuned to deliver the optimal policy of the real system even when using a wrong model, and one practical outcome of the theory proposed in this paper is that all RL techniques can be directly used to tune the NMPC scheme to increase its performance on the real system, and this was the first work proposing to use NMPC as a function approximator in RL. In [27] Gros and Zanon proposed an RL formulation based on MPC which addresses the issue of safety, which they did not rigorously enforce in [28, 59]. A key idea is that MPC is used as a function approximator within RL to provide safety and stability guarantees; and RL is used to tune the MPC parameters, thus improving closed-loop performance in a data-driven fashion [27, 28, 59].

Recently, a method to implement the stochastic policy gradient method using actor-critic techniques was proposed in [29], where the policy is approximated using an MPC scheme, and a computationally inexpensive approach is used to build a stochastic policy generating samples that are supposed to be feasible for the MPC constraints. In [60], a novel algorithm called reinforced predictive control (RL-MPC) that merges the relative merits of MPC and RL, and the complementarity between RL and MPC is emphasized.

In summary, the integration of MPC and RL can provide a more efficient, robust, and adaptive control system for autonomous driving.

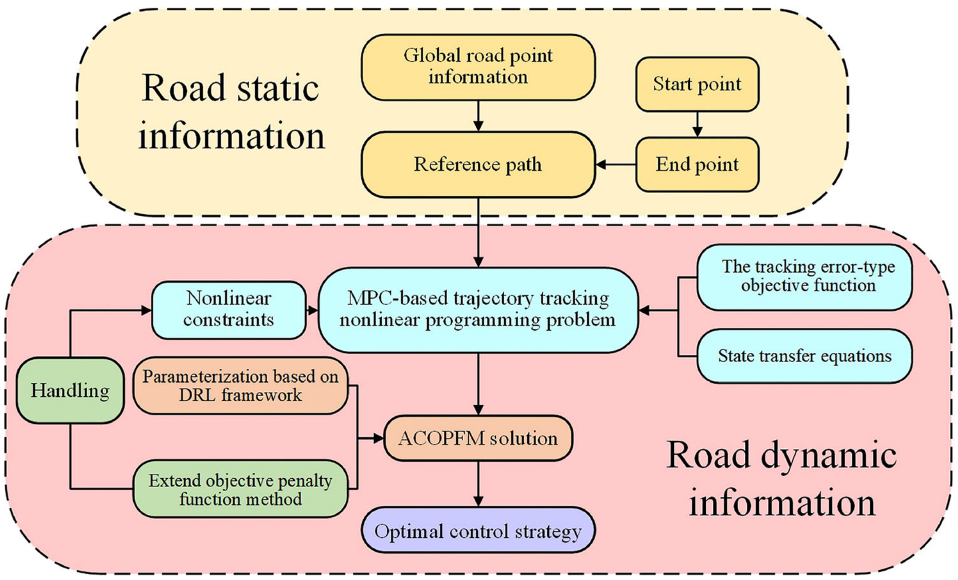

Road static information
Global road point information
Start point
Reference path
End point
Nonlinear constraints
MPC-based trajectory tracking nonlinear programming problem
The tracking error-type objective function
State transfer equations
Handling
Parameterization based on DRL framework
ACOPFM solution
Extend objective penalty function method
Optimal control strategy
Road dynamic information

Fig. 1 Illustration of our integrated decision and control scheme

Planning and control scheme

This section describes our planning and control scheme employed based on the IDC framework [30], which consists of two main modules: path planning and trajectory tracking control, as seen in Fig. 1.

Firstly, the global road points are generated in consideration of the road static information [30], including road characteristics (shapes, intersections, obstacles etc.) and traffic rules (traffic light signals, vehicle speed limits etc.), then the reference paths are filtered using the static information from the starting point to the end point of the ego vehicle, which are used as the underlying reference paths for the dynamic optimal tracking task (in the CARLA simulator, the built-in A^* algorithm can quickly generate the reference paths). Next, use the dynamic traffic information including surrounding vehicles' movements to formulate the constrained trajectory tracking problem with respect to each candidate path, solve the formulated problems separately, and follow the path with the best tracking performance.

The formulated MPC-based trajectory tracking problem consists of three parts: first, the objective function to evaluate the trajectory tracking effect; second, the state transfer equation constraints based on the bicycle vehicle model [14] and vehicle kinematics for the ego and other vehicles, respectively; third, the constraints resulting from dynamic collision avoidance in consideration of the surrounding vehicles, and the distance restrictions in consideration of the lane boundaries.

For the constrained optimization problem faced, the ACOPFM is developed as the solution scheme. The objective penalty function method in the optimization field is employed to handle the constraints in order to obtain the unconstrained optimization problem and further the loss function required for the DNN model training, and the objective penalty parameters are adjusted adaptively during the model training process to avoid the ill-conditioning issue.

As seen in Fig. 2, the highway scenario is used to demonstrate the trajectory tracking task. The A^* algorithm is used to generate the reference paths. The optimal trajectory for the ego vehicle is obtained by the ACOPFM. Note that the goal of the optimization problem is to minimize the trajectory tracking error between the trajectory of the ego vehicle and the reference trajectory as much as possible under the premise of safety and reasonableness.

MPC-based nonlinear programming problem

The trajectory tracking model for autonomous driving was given in [30]. Equation(1) shows the variables related to the state and control.

$$\begin{aligned} \boldsymbol{x}_{i|t} &= [l_{\mathrm{x}},\ l_{\mathrm{y}},\ v_{\mathrm{lon}},\ v_{\mathrm{lat}},\ \varphi,\ \omega]_{i|t}^{\top} \\ \boldsymbol{x}_{i|t}^{j} &= [l_{\mathrm{x}}^{j},\ l_{\mathrm{y}}^{j},\ v_{\mathrm{lon}}^{j},\ 0,\ \varphi^{j},\ 0]_{i|t}^{\top} \\ \boldsymbol{x}_{i|t}^{\mathrm{lane}} &= [l_{\mathrm{x}}^{\mathrm{lane}},\ l_{\mathrm{y}}^{\mathrm{lane}},\ 0,\ 0,\ 0,\ 0]_{i|t}^{\top} \\ \boldsymbol{x}_{i|t}^{\mathrm{ref}} &= [l_{\mathrm{x}}^{\mathrm{ref}},\ l_{\mathrm{y}}^{\mathrm{ref}},\ v_{\mathrm{lon}}^{\mathrm{ref}},\ 0,\ \varphi^{\mathrm{ref}},\ 0]_{i|t}^{\top} \\ \boldsymbol{u}_{i|t} &= [\delta,\ a]_{i|t}^{\top} \end{aligned} \tag{1}$$

The prediction horizon is set as T steps. $x_{i|t}$ is the state of the ego vehicle at the ith time step within T from the current time step t. $x_{i|t}^{j}$ is the state of the jth vehicle in $\boldsymbol{N}$, which stands for a collection of vehicles interacting with the ego vehicle, where l_{x} and l_{y} are the position coordinates (for ego and other vehicles, they stand for the position of the center

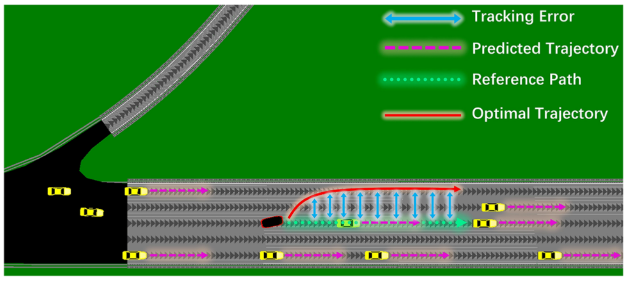


Fig. 2 Schematic diagram of the trajectory tracking task on highway scenario. The blue line segment illustrates the trajectory tracking error. The pink dashed segment illustrates the predicted trajectory of other vehicles, according to the state transfer equation. The red thick line segment is the optimized trajectory obtained for the ego vehicle after dynamically solving the MPC-based trajectory tracking problem using ACOPFM. The green solid line is the reference path, generated by the A^* algorithm in the CARLA simulator

of gravity(CG)). v_{lon} and v_{lat} are the longitudinal and lateral velocities, φ is the heading angle, ω is the yaw rate. $x_{i|t}^{\text{ref}}$ and $x_{i|t}^{\text{lane}}$ are the closest points from $x_{i|t}$ on the reference path and on the road edge. $u_{i|t}$ is the control of the ego vehicle at the ith time step within T from the current time step t. The vehicle control is expressed in terms of the front-wheel angle δ and acceleration a.

The objective function is given in (2):

$$\min_{u_{i|t}, i=0:T-1} \boldsymbol{J}(\boldsymbol{x}_{i|t}, \boldsymbol{u}_{i|t}) = \sum_{i=0}^{T-1} \boldsymbol{l}(\boldsymbol{x}_{i|t}) + \boldsymbol{\eta}(\boldsymbol{u}_{i|t}) \tag{2}$$

where

$$\begin{aligned} \boldsymbol{l}(\boldsymbol{x}_{i|t}) &= ||\boldsymbol{x}_{i|t} - \boldsymbol{x}_{i|t}^{\text{ref}}||_2^2 \\ &= (\boldsymbol{x}_{i|t} - \boldsymbol{x}_{i|t}^{\text{ref}})^\top \boldsymbol{Q}(\boldsymbol{x}_{i|t} - \boldsymbol{x}_{i|t}^{\text{ref}}), \\ \boldsymbol{\eta}(\boldsymbol{u}_{i|t}) &= \boldsymbol{u}_{i|t}^\top \boldsymbol{R}\boldsymbol{u}_{i|t}, \end{aligned}$$

$\boldsymbol{Q}$, $\boldsymbol{R}$ are positive definite weighting matrices. Note that the trajectory tacking problem is formulated as a rolling optimization process where for each step the new $\boldsymbol{x}_{0|t}$ is provided, then the tracking error is minimized to get the optimal control sequence $\{\boldsymbol{u}_{0|t}, \boldsymbol{u}_{1|t}, ...\boldsymbol{u}_{T-1|t}\}$. The actual control given to the vehicle is $\boldsymbol{u}_{0|t}$ each time due to the uncertainty of the dynamic road information. The state of the vehicle at the next time step is obtained by the state transfer equation. This optimization process is repeated at each time step [8, 61]. Based on the vehicle kinematics and the bicycle vehicle model [14], the state transfer equation of ego vehicle is

$$\boldsymbol{x}_{i+1|t} = \boldsymbol{F}_{ego}(\boldsymbol{x}_{i|t}, \boldsymbol{u}_{i|t}),$$

which is specifically given in Eq. (3):

$$\begin{bmatrix} v_{\text{lon}|i+1} \\ \varphi_{i+1} \\ l_{\text{x}|i+1} \\ l_{\text{y}|i+1} \\ v_{\text{lat}|i+1} \\ \omega_{i+1} \end{bmatrix} = \begin{bmatrix} v_{\text{lon}|i} + T_s \cdot a_i \\ \dfrac{\tan \delta_i}{L} \cdot \varphi_i + T_s \cdot v_{\text{lon}|i+1} \\ l_{\text{x}|i} + T_s \cdot v_{\text{lon}|i+1} \cdot \cos \varphi_{i+1} \\ l_{\text{y}|i} + T_s \cdot v_{\text{lon}|i+1} \cdot \sin \varphi_{i+1} \\ v_{\text{lat}|i} \\ \omega_i \end{bmatrix} \tag{3}$$

The state transfer equation of other vehicles is

$$\boldsymbol{x}_{i+1|t}^j = \boldsymbol{F}_{other}(\boldsymbol{x}_{i|t}^j),$$

which is specifically given in nonlinear Eq.(4):

$$\begin{bmatrix} v_{\text{lon}|i+1}^j \\ \varphi_{i+1}^j \\ l_{\text{x}|i+1}^j \\ l_{\text{y}|i+1}^j \\ v_{\text{lat}|i+1}^j \\ \omega_{i+1}^j \end{bmatrix} = \begin{bmatrix} v_{\text{lon}|i}^j \\ \varphi_i^j \\ l_{\text{x}|i}^j + T_s \cdot v_{\text{lon}|i+1}^j \cdot \cos \varphi_{i+1}^j \\ l_{\text{y}|i}^j + T_s \cdot v_{\text{lon}|i+1}^j \cdot \sin \varphi_{i+1}^j \\ v_{\text{lat}|i}^j \\ \omega_i^j \end{bmatrix} \tag{4}$$

Next we move to the nonlinear constraints involved in the model:

$$\begin{aligned} ||\boldsymbol{x}_{i|t} - \boldsymbol{x}_{i|t}^j||_2 &\geq \boldsymbol{D}_{other}^{\text{safe}}, \\ ||\boldsymbol{x}_{i|t} - \boldsymbol{x}_{i|t}^{\text{lane}}||_2 &\geq \boldsymbol{D}_{lane}^{\text{safe}}, \end{aligned}$$

where $\boldsymbol{D}_{other}^{\text{safe}}$ is the safe distance between the ego vehicle and other vehicles, while $\boldsymbol{D}_{lane}^{\text{safe}}$ is the safe distance between the ego vehicle and the lane line. More specifically, we have the following nonlinear constraints, as shown in Eq. (5), which update dynamically at each time step with the change of the ego vehicle's state, other vehicles' states, lane boundary information, and number of other vehicles:

$$\begin{cases} \sqrt{(l_{\text{x}} - l_{\text{x}}^j)^2 + (l_{\text{y}} - l_{\text{y}}^j)^2} \geq \boldsymbol{D}_{other}^{\text{safe}}, \ j \in \boldsymbol{N} \\ \sqrt{(l_{\text{x}} - l_{\text{x}}^{\text{lane},1})^2 + (l_{\text{y}} - l_{\text{y}}^{\text{lane},1})^2} \geq \boldsymbol{D}_{lane}^{\text{safe}} \\ \sqrt{(l_{\text{x}} - l_{\text{x}}^{\text{lane},2})^2 + (l_{\text{y}} - l_{\text{y}}^{\text{lane},2})^2} \geq \boldsymbol{D}_{lane}^{\text{safe}} \end{cases} \tag{5}$$

where $l^{\text{lane},1}$ and $l^{\text{lane},2}$ represent the left lane boundary and the right lane boundary.

A model predictive control-based nonlinear programming problem is given as follows to fully describe the trajectory tracking problem:

$$\begin{aligned}\min_{u_{i|t}, i=0:T-1} \; & \boldsymbol{J}(\boldsymbol{x}_{i|t}, \boldsymbol{u}_{i|t}) = \sum_{i=0}^{T-1} \boldsymbol{l}(\boldsymbol{x}_{i|t}) + \boldsymbol{\eta}(\boldsymbol{u}_{i|t}) \\ \text{s.t.} \; & \boldsymbol{x}_{i+1|t} = \boldsymbol{F}_{ego}(\boldsymbol{x}_{i|t}, \boldsymbol{u}_{i|t}) \\ & \boldsymbol{x}^j_{i+1|t} = \boldsymbol{F}_{other}(\boldsymbol{x}^j_{i|t}) \\ & ||\boldsymbol{x}_{i|t} - \boldsymbol{x}^j_{i|t}||_2 \geq \boldsymbol{D}^{\text{safe}}_{other} \\ & ||\boldsymbol{x}_{i|t} - \boldsymbol{x}^{\text{lane}}_{i|t}||_2 \geq \boldsymbol{D}^{\text{safe}}_{lane} \\ & \boldsymbol{x}_{0|t} = \boldsymbol{x}_t, \boldsymbol{x}^j_{0|t} = \boldsymbol{x}^j_t, \boldsymbol{u}_{0|t} = \boldsymbol{u}_t \\ & i = 0 : T-1, \; j \in \boldsymbol{N}. \end{aligned} \tag{6}$$

As the problem is highly nonlinear, the number of constraints involved is large, and the vehicle-mounted computing resources are limited, the traditional numerical optimization methods are not able to solve the problem satisfactorily. It has been shown that the model-based RL algorithms can produce reasonable solutions for such problems [30, 62]. However, the constraints involved pose a great challenge to these algorithms. In the next section, we will present a generalized objective penalty function method to handle the constraints under the model-based RL framework, which can well handle the ill-conditioning issue that occurs in the traditional penalty function methods.

ACOPFM framework

DRL-based parameterization

We consider a standard RL setup for the interaction of the agent with the environment in discrete time steps. Here the agent is defined as an autonomous vehicle and the simulation environment can be modeled as a Markov decision process (MDPs), which can be defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \mathcal{V})$. The state space and action space are identified as being continuous. Within each discrete time step, for each state given by the simulation environment $s_t \in \mathcal{S}$, the ego vehicle takes the appropriate action $a_t \in \mathcal{A}$ according to the input state s_t and the policy $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ to complete the trajectory tracking task. The function $r(s_t, a_t) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ describes the cost, indicating the objective function of the trajectory tracking task optimized at each moment, which is represented here as $l(x_t) + \eta(u_t)$. The function $p(s_{t+1}|s_t, a_t) : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(s_{t+1})$ represents the probability density of the next state s_{t+1} to which the current state s_t is transferred by taking action a_t. $p(\cdot)$ is usually unknown due to the complexity and uncertainty of the autonomous vehicle dynamics and the motion of the surrounding traffic participants, but we can obtain the state at the next time step according to the state transfer equation $s_{t+1} = \boldsymbol{F}(s_t)$. The state value function $V^\pi(s_t)$ is defined as the expected cost sum from the start of s_t to future moments and can be used to evaluate the value of an initial state s_t under policy π. It is closely related to the policy π of the selected action by $V^\pi(s_t) = \left\{ \sum_{t=0}^{T-1} l(x_t) + \eta(u_t) | s_0 = s_t \right\}$.

We use the actor-critic RL scheme to solve the trajectory tracking problem. The purpose of actor is to find the optimal policy π^*_θ, which demonstrates the tracking performance during the training of NNs. The purpose of critic is to approximate the true function value, and to improve the closed-loop performance as the training of NNs becomes more and more accurate, the loss of critic can demonstrate the safety performance during the training process.

Our RL problems are modeled as follows:

$$\begin{aligned} \min_\theta J_{actor} &= \mathbb{E}_{s_{0|t}} \left\{ \sum_{i=0}^{T-1} \boldsymbol{l}(s_{i|t}, \pi_\theta(s_{i|t})) + \boldsymbol{\eta}(u_{i|t}, \pi_\theta(s_{i|t})) \right\} \\ \text{s.t. } & s_{i+1|t} = \boldsymbol{F}(s_{i|t}, \pi_\theta(s_{i|t})) \\ & g_c(s_{i|t}) \geq 0, c \in \boldsymbol{C} \\ & s_{0|t} = s_t \leftarrow \{\tau^*, x_t, x^j_t, j \in N\} \sim d \\ & i = 0 : T-1 \end{aligned} \tag{7}$$

$$\begin{aligned} & \min_\omega J_{critic} \\ &= \mathbb{E}_{s_{0|t}} \left\{ \left(\sum_{i=0}^{T-1} \boldsymbol{l}(s_{i|t}, \pi_\theta(s_{i|t})) + \boldsymbol{\eta}(u_{i|t}, \pi_\theta(s_{i|t})) - \boldsymbol{V}_\omega(s_{0|t}) \right)^2 \right\} \\ & \text{s.t. } s_{i+1|t} = \boldsymbol{F}(s_{i|t}, \pi_\theta(s_{i|t})) \\ & s_{0|t} = s_t \leftarrow \{\tau^*, x_t, x^j_t, j \in N\} \sim d \\ & i = 0 : T-1, \end{aligned} \tag{8}$$

where the tracking error objective

$$\begin{aligned} & \boldsymbol{l}(s_{i|t}, \pi_\theta(s_{i|t})) + \boldsymbol{\eta}(u_{i|t}, \pi_\theta(s_{i|t})) \\ & = ||\boldsymbol{x}_{i|t} - \boldsymbol{x}^{\text{ref}}_{i|t}||^2_2 + \boldsymbol{\pi}_\theta(s_{i|t})^\top \boldsymbol{R} \boldsymbol{\pi}_\theta(s_{i|t}), \end{aligned}$$

$g_c(s_{i|t}) \geq 0, c \in \boldsymbol{C}$ denote all the constraints. As the number of constraints is dynamically changing according to the surrounding vehicle information and the location information at each moment. Note that $\boldsymbol{F} \in \{\boldsymbol{F}_{ego}, \boldsymbol{F}_{other}\}$. In the value network, the optimal value function $\boldsymbol{V}_{\omega^*}$ is obtained by fitting the optimization parameter ω. We train the value network so that $\boldsymbol{V}_{\omega^*}$ approximates the obtained $\boldsymbol{J^*}$. The control $u_{i|t} = \pi_\theta(s_{i|t})$ can be obtained via the policy network. Under some assumptions, $u_{i|t}$ can be mapped as $\pi_\theta(s_{i|t})$, and the optimal policy $\pi^*_\theta(s_{i|t})$is equivalent to optimal action $u^*_{i|t}$.

Actor-critic objective penalty function method

Taking $P(t) = \max\{0, t\}^2$ and introducing a dynamic objective penalty parameter M, then the constrained critic problem is converted to a generalized objective penalty function problem as follows:

$$
\begin{aligned}
\min_{\theta} J_p &= Q(J_{actor} - M) + J_{penalty} \\
&= \mathbb{E}_{s_{0|t}}\left\{Q\left(\sum_{i=0}^{T-1} \boldsymbol{l}(s_{i|t}, \pi_\theta(s_{i|t})) + \boldsymbol{\eta}(u_{i|t}, \pi_\theta(s_{i|t})) - M_\theta\right)\right\} \\
&\quad + \mathbb{E}_{s_{0|t}}\left\{\sum_{i=0}^{T-1} \phi_i(\theta)\right\} \\
&\text{s.t. } s_{i+1|t} = \boldsymbol{F}(s_{i|t}, \pi_\theta(s_{i|t})) \\
&\phi_i(\theta) = \sum_{c \in \boldsymbol{C}} [\max\{0, -g_c(s_{i|t})\}]^2 \\
&s_{0|t} = s_t \leftarrow \{\tau^*, x_t, x_t^j, j \in J\} \sim d \\
&i = 0 : T-1.
\end{aligned} \tag{9}
$$

The algorithmic procedure of the actor-critic objective penalty method is given in Algorithm 1. Let the set

$$U = \{u = \pi_\theta(s_t) \,|\, g_c(s_t) \geq 0, c \in \boldsymbol{C}\}.$$

The optimal control taken each time through the policy network is u_k. $f(u_k)$ is the value obtained from $\sum_{i=0}^{T-1} \boldsymbol{l}(s_{i|t}, \pi_\theta(s_{i|t})) + \boldsymbol{\eta}(u_{i|t}, \pi_\theta(s_{i|t}))$. Since the punishment for constraint violations is formulated as a penalty term, u_k does not necessarily satisfy the constraints.

Algorithm 1 ACOPFM for Offline Training

```
Initialize: Policy network $\pi_\theta$ and value network $V_\omega$ with random paramaters $\theta$ and $\omega$,
  buffer $\mathcal{B} \leftarrow \emptyset$, the learning rates $\alpha_\theta$ and $\alpha_\omega$.
  for each iteration do
    // Sampling (from the environment)
    Select a optimal candidate path $\tau^* \subset \Pi$,
    initialize ego vehicle state $x_t$ and other vehicles states $x_t^j$, $j \in \mathbf{N}$.
    for each environment step do
      $s_t \leftarrow \{\tau^*, x_t, x_t^j, j \in \mathbf{N}\}$ ,
      $\mathcal{B} \cup \{s_t\}$ ,
      $u_t = \pi_\theta(s_t)$ ,
      Apply $u_t$ to ego vehicle and other vehicles to observe $x_{t+1}, x_{t+1}^j$, $j \in \mathbf{N}$.
    end for
    // Optimizing (ACOPFM)
    Iteration counter $k \leftarrow 0$.
    Initialize parameters $\theta_0$ and $\omega_0$.
    Apply $f$ and $\pi_\theta$ to compute $J_{critic}$ and $J_p$.
    // Updating the value network parameter
    Apply value network $V_\omega$ to update:
    $\omega \rightarrow \omega - \alpha_\omega \nabla_\omega J_{critic}$.
    // Updating the policy network parameter
    if $k = 0$ then
      Randomly choose $a_0$ to satisfy $a_0 < 0$,
      choose $u_0 \in U$ to satisfy $J_{penalty}(\theta_0) = 0$.
      Let $b_0 = f(u_0)$, $M_0 = \frac{a_0 + b_0}{2}$.
      Apply policy network $\pi_0$ to update:
      $\theta_1 \rightarrow \theta_0 - \alpha_\theta \nabla_\theta J_{actor}(\theta_0, M_0)$.
    else
      Apply policy network $\pi_\theta$ to observe:
      $f(u_k)$, $J_p(\theta_k, M_{k-1})$, $J_{penalty}(\theta_k)$.
      if $J_p(\theta_k, M_{k-1}) = 0$ then
        Let $a_k = a_{k-1}$, $b_k = f(u_k)$, $M_k = \frac{a_k + b_k}{2}$.
        Apply policy network $\pi_\theta$ to update:
        $\theta_{k+1} \rightarrow \theta_k - \alpha_\theta \nabla_\theta J_{actor}(\theta_k, M_k)$.
      else
        if $J_{penalty}(\theta_k) > 0$ then
          Let $a_k = \max\{f(u_k), M_{k-1}\}$, $b_k = b_{k-1}$, $M_k = \frac{a_k + b_k}{2}$.
          Apply policy network $\pi_\theta$ to update:
          $\theta_{k+1} \rightarrow \theta_k - \alpha_\theta \nabla_\theta J_{actor}(\theta_k, M_k)$.
        else
          Apply policy network $\pi_\theta$ to update:
          $\theta_{k+1} \rightarrow \theta_k - \alpha_\theta \nabla_\theta J_{actor}(\theta_k, M_k)$.
        end if
      end if
    end if
  end for
```

Convergence analysis of the ACOPFM

We will show that the ACOPFM converges to the optimal policy under certain conditions, and we say that a "round" is completed when an optimization process is completed.

Assumption 1 After the round k completes, we have an optimized policy parameter θ_k. Let

$$P(\theta_k, \beta) = \left\{\theta_k \,\middle|\, J_{actor}(\theta_k) \leq \beta, k = 1, 2, ...\right\}, \tag{10}$$

which is called a generalized P-level set. We assume that $P(\theta_k, \beta)$ is bounded for any given $\beta > 0$ if the sequence $\{M_k\}$ is convergent.

Next we establish the convergence of the sequence $\{M_k\}$.

Theorem 1 *Let $M_* = \min_{\theta} J_{actor}(\theta)$, then $M_* = f(\pi_{\theta^*}(s_t))$. Suppose that for some M, u_M is the control obtained from $J_p(\theta, M)$, and let θ_M be the optimal parameter corresponding to $J_p(\theta, M)$, i.e. $u_M = \pi_{\theta_M}(s_t)$. Then the following three assertions hold.*

(i) *If $J_p(\theta_M, M) = 0$, then $u_M \in U$ and $M_* \leq f(u_M) \leq M$.*
(ii) *If $J_p(\theta_M, M) > 0$ and $u_M \notin U$, then $M \leq M_*$ and $f(u_M) < M_*$.*
(iii) *If $J_p(\theta_M, M) > 0$ and $u_M \in U$, then $\theta_M \in \arg\min_{\theta} J_{actor}(\theta)$.*

Proof **(i)** The conclusion clearly holds according to the definitions of P and Q. **(ii)** Assume that $\theta^* \in \arg\min_{\theta} J_{actor}(\theta)$.

Obviously

$$J_{penalty}(\theta^*) = 0, J_{actor}(\theta^*) = f(\pi_{\theta^*}(s_t)).$$

As $J_p(\theta_M, M) > 0$, it follows

$$0 < J_p(\theta_M, M) \leq J_p(\theta^*, M) = Q(f(\pi_{\theta^*}(s_t) - M)).$$

By the definition of Q, $M < f(\pi_{\theta^*}(s_t)) = M_*$.

If $f(u_M) \leq M$, then $f(u_M) \leq M \leq M_*$. If $f(u_M) > M$, then

$$\begin{aligned} 0 &< Q(f(u_M) - M) \\ &\leq J_p(\theta_M, M) \leq J_p(\theta^*, M) = Q(f(\pi_{\theta^*}(s_t) - M)). \end{aligned}$$

Therefore, $f(u_M) < f(\pi_{\theta^*}(s_t)) = M_*$. At the initial step 0, the ego vehicle is generated at the starting point, the default parameter settings satisfy the constraints, therefore $J_{penalty}(\theta_0) = 0$.

(iii) According to the given conditions, we have

$$\begin{aligned} 0 &< Q(f(u_M) - M) \\ &= J_p(\theta_M, M) \leq J_p(\theta, M) \\ &= Q(f(u) - M), \forall u \in U. \end{aligned}$$

Since $u_M \in U$, this implies that

$$f(u_M) - M \leq f(u) - M, \forall u \in U,$$

i.e. $\theta_M \in \arg\min_{\theta} J_{actor}(\theta)$. □

Theorem 2 *Let $\{M_k\}$ be the sequence of objective penalty parameters generated during the iteration with the sequence $\{a_k\}$ and the sequence $\{b_k\}$. Then $\{a_k\}$ is an increasing sequence and $\{b_k\}$ is a decreasing sequence and satisfies*

$$a_k \leq M_* \leq b_k, \qquad k = 1, 2, ..., \tag{11}$$

$$b_{k+1} - a_{k+1} \leq \frac{b_k - a_k}{2}, \qquad k = 1, 2, ... \tag{12}$$

Proof Considering the case when $k = 0$. Clearly, $a_0 \leq M_* \leq b_0$. When $k = 1$, if $J_p(\theta_1, M_0) = 0$, then $a_1 = a_0,\ b_1 = f(u_1) = M_0,\ M_1 = \frac{a_1 + b_1}{2}$, thus we have

$$b_1 - a_1 = M_0 - a_0 = \frac{a_0 + b_0}{2} - a_0 = \frac{b_0 - a_0}{2};$$

else $J_p(\theta_1, M_0) \neq 0$, then for $J_{penalty}(\theta_k) = 0$, it holds, for the case $J_{penalty}(\theta_k) > 0$,

$$\begin{aligned} a_1 &= \max\{f(u_1), M_0\} \\ &\geq M_0 = \frac{a_0 + b_0}{2} \geq \frac{a_0 + a_0}{2} = a_0. \end{aligned}$$

$$b_1 = b_0.$$

thus

$$\begin{aligned} b_1 - a_1 &= b_0 - \max\{f(u_1), M_0\} \\ &\leq b_0 - M_0 = b_0 - \frac{a_0 + b_0}{2} = \frac{b_0 - a_0}{2}, \end{aligned}$$

this implies that $b_1 - a_1 \leq \frac{b_0 - a_0}{2}$.

Considering the case $k \geq 1$. By induction, suppose that $a_k \leq M_* \leq b_k$ for some $k \geq 1$, then we consider the case of $k + 1$.

If $J_p(\theta_{k+1}, M_k) = 0$, this implies that $u_{k+1} \in U$, $f(u_{k+1}) = M_k = b_{k+1}$, $a_{k+1} = a_k$, $M_{k+1} = \frac{a_{k+1} + b_{k+1}}{2}$, thus

$$a_{k+1} = a_k \leq M_* \leq b_{k+1} = M_k = \frac{a_k + b_k}{2} \leq b_k.$$

$$b_{k+1} - a_{k+1} = M_k - a_k = \frac{b_k - a_k}{2}.$$

If $J_p(\theta_{k+1}, M_k) \neq 0$, then for $J_{penalty}(\theta_{k+1}) = 0$, obviously it holds. For the case $J_{penalty}(\theta_{k+1}) > 0$, we have $b_{k+1} = b_k,\ a_{k+1} = \max\{f(u^{k+1}), M_k\} \geq M_k$, thus

$$a_{k+1} \geq M_k = \frac{a_k + b_k}{2} \geq \frac{a_k + a_k}{2} = a_k.$$

□

From **Theorem 1**, we have

$$a_{k+1} \leq M_* \leq b_k = b_{k+1}.$$

Therefore,

$$\begin{aligned} b_{k+1} - a_{k+1} &= b_k - \max\{f(u^{k+1}), M_k\} \\ &\leq b_k - M_k = \frac{b_k - a_k}{2}. \end{aligned}$$

Also, we can find that the sequence $\{a_k\}$ is increasing and the sequence $\{b_k\}$ is decreasing, and both of them are bounded. Therefore, the sequences $\{a_k\}$ and $\{b_k\}$ are both convergent. Let

$$\lim_{k\to\infty} a_k = a^*, \ \lim_{k\to\infty} b_k = b^*.$$

By Eqs. (11) and (12), it is clear that $a^* = b^*$. Then we have that the sequence $\{M_k\}$ converges to $a^* = b^*$.

Theorem 3 *Under **Assumption 1**, the limit point of any convergent subsequence of the sequence of the optimization parameters $\{\theta_k\}, k = 1, 2...,$ is the optimal solution to the original problem, where each θ_k is obtained in the kth iteration.*

Proof We first show that the sequence of optimization parameters $\{\theta_k\}, k = 1, 2...$ is bounded.

Since θ_k is the optimal parameter after k rounds of iterations and $u_0 \in U$, $J_{penalty}(\theta_0) = 0$ when $k = 0$, we have

$$J_p(\theta_k, M_{k-1}) \le Q(J_{actor}(\theta_0) - M_{k-1}), \quad k = 0, 1, 2, ...$$

by the convergence of $\{M_k\}$, i.e. $\lim_{k\to\infty} M_k = a^*$.

By **Assumption 1**, there exists $\overline{\beta} > 0$, when $\beta > \overline{\beta}$ we have

$$\begin{aligned} J_p(\theta_k, M_{k-1}) &\le Q(J_{actor}(\theta_0) - M_{k-1}) \\ &< \beta. \quad k = 0, 1, 2, ... \end{aligned}$$

as P-level set $P(\theta_k, \beta)$ is bounded, i.e. the sequence $\{\theta_k\}$ is bounded.

Since $M_* = \min_\theta J_{actor}(\theta)$, without loss of generality, let $\theta^* \in \arg\min_\theta J_{actor}(\theta)$. We have proved that $a_k \le M_* \le b_k, k = 0, 1, 2...$ and the sequences $\{a_k\}, \{b_k\}, \{M_k\}$ converge to a^*. Let $k \to +\infty$, we obtain $a^* = M_* = J_{actor}(\theta^*)$.

Note that J_p, J_{actor}, $J_{penalty}$ are all continuous on the parameter space. As the sequence θ_k is bounded, it has convergent subsequence $\{\theta_{k_j}\}$ with $\theta_{k_j} \to \overline{\theta}$ as $k_j \to +\infty$.

We will show that $J_{actor}(\overline{\theta}) = a^*$. Clearly $M_* = J_{actor}(\theta^*)$, and note that

$$\begin{aligned} J_p(\theta_{k_j}, M_{k_j-1}) &\le J_p(\theta^*, M_{k_j-1}) \\ &= Q(J_{actor}(\theta^*) - M_{k_j-1}). \end{aligned}$$

Let $k_j \to +\infty$, we have $M_{k_j-1} \to a^*$. Then

$$\begin{aligned} \lim_{k_j\to+\infty} J_p(\theta_{k_j}, M_{k_j-1}) &= J_p(\overline{\theta}, a^*) \\ &\le Q(J_{actor}(\theta^*) - a^*) \\ &= 0. \end{aligned}$$

Therefore $J_p(\overline{\theta}, a^*) \le 0$. It is clear that $J_p(\overline{\theta}, a^*) = 0$ due to the nonnegativity, and $J_{penalty}(\overline{\theta}) = 0$, $J_{actor}(\overline{\theta}) = a^*$, i.e., the limit of any convergent subsequence of the parameter sequence $\{\theta_k\}, k = 0, 1, 2...$ is the optimal solution. □

Simulation experiments

Our simulation experiments are conducted on the CARLA-SUMO joint platform. In order to simulate the real-time trajectory tracking scenario under complex constraints, the simulation platform employed should have realistic sensor simulation, vehicle dynamics simulation, and traffic flow simulation. We thus choose CARLA, which comes with a realistic sensor simulation system, a vehicle dynamics simulation system, and a game engine rendering [63]. SUMO, which has realistic complex random traffic simulation [64], is chosen to produce realistic traffic flow in CARLA, and some important variable settings in SUMO simulation traffic flow are shown in Table 1. For the experimental scenario, we choose the multi-lane setting of the CARLA 09.13 version of the Town06 map, and it is worth noting that the vehicle speed limit is set at 20 m/s.

Table 1 Important variable settings in SUMO simulation traffic flow

Variable	Meaning	Value
Begin	Start time	0 s
End	End time	999999 s
DepartLane	Generate lanesfor the vehicle	Random (1–5)
DepartPos	Generate the initial vehicle position	Random
DepartSpeed	Vehicle initial speed	Random
VehsPerHour	Vehicle generation frequency	Random

Experimental environment

We choose the $vehicle.tesla.model3$ simulation model in CARLA simulator as ego vehicle for the trajectory tracking task. The state information is known to contain three pieces: ego vehicle's state information, other vehicles' state information, and information about the reference state:

$$s = [s^{\text{ego}}, s^{\text{other}}, s^{\text{ref}}].$$

The specific settings of s^{ego} are described in Sect. "Design of the nonlinear programming problem". $s^{\text{other}} = [l_{\text{x}}^j, l_{\text{y}}^j, v_{\text{lon}}^j, \varphi^j]_{i|t}^T, j \in \boldsymbol{N}$, and $\boldsymbol{N} = 6$. $s^{\text{ref}} = [l_{\text{x}}^{\text{ref}}, l_{\text{y}}^{\text{ref}}, v_{\text{lon}}^{\text{ref}}, \varphi^{\text{ref}}]_{i|t}^\top$, in which

$$(l_{\text{x}}^{\text{ref}},\ l_{\text{y}}^{\text{ref}}) \in \arg\min_i \left\{\sqrt{(l_{\text{x}} - l_{\text{x}}^{\text{ref}})^2 + (l_{\text{y}} - l_{\text{y}}^{\text{ref}})^2}\right\}.$$

The reference speed $v_{\text{lon}}^{\text{ref}}$is $13m/s$, which is generated based on the real time traffic rules, and the reference heading angle φ^{ref} is generated adaptively by the CARLA simulator. In the policy network, the input space of the DNN contains information on all the three states, which is a 34-dimensional tensor.

The control is set as

$$u = [\delta,\ a]^\top,$$

where δ is the front-wheel angle, and can be adjusted by changing the steering wheel angle in the CARLA simulator (the steering wheel angle range is $[-1,\ 1]rad$), a is the acceleration of the ego vehicle, and can be converted to throttle in the CARLA simulator by the PID controller, parameterized by the throttle range $[0,\ 1]$. The output space contains

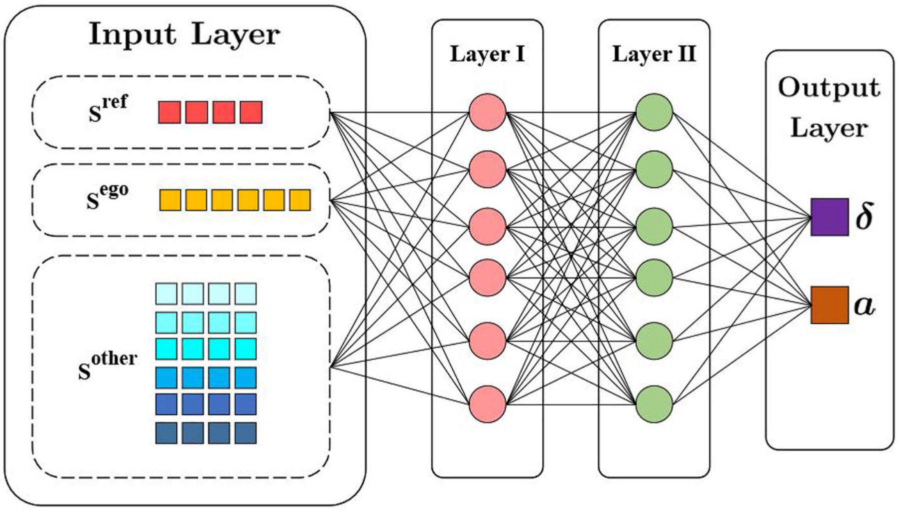


Fig. 3 Schematic diagram of the policy network structure. The input layer has 34 neurons, i.e., the input space is 34-dimensional. They include the ego vehicle information (6 dimensions), the 6 other vehicles information (6×4 dimensions) and the reference information (4 dimensions). The number of neurons in both hidden layers is 256, and ELU is used as the activation function. To ensure that the control of the network output conforms to the boundary constraints of the control, a sigmoid function with parameters is used to map the network output to the corresponding intervals, and the CARLA simulator simulates the control of the autonomous vehicle by adjusting the steering wheel angle and throttle size accordingly

the angle and acceleration of the front wheel, which is a 2-dimensional tensor with an angle range $[-0.4,\ 0.4]rad$, and acceleration range $[-3,\ 3]m/s^2$. Due to the nonlinear mapping relationship, we use ELU function to activate the network output and use sigmoid function with parameters to map the network output to appropriate intervals in order to ensure that the control of the network output is in line with the control range. The basic structure of the policy network is shown in Fig. 3. The value network is also a DNN with two hidden layers, but it differs structurally from the policy network in that the value network outputs a real-valued signal. This real-valued signal is the score of the state-control pair.

The information of all road points of the map is obtained from CARLA simulation platform in advance. During the simulation process, the ego vehicle is automatically destroyed when it reaches the end point, collides with other vehicles, or crosses the boundary of the lane, and then it is generated again at the starting position. The data of the start and end coordinates of the reference path are shown in Table 2.

The prediction horizon $T = 30$, and the simulation interval time step T_s for each step is $0.05s$. The positive definite weighting matrix is

$$\boldsymbol{Q} = diag\{0.04, 0.04, 0.01, 0.05, 0.1, 0.06\},$$
$$\boldsymbol{R} = diag\{0.1, 0.05\}.$$

Safe distance $\boldsymbol{D}_{other}^{\text{safe}} = 3.0\,m$, $\boldsymbol{D}_{lane}^{\text{safe}} = 1.5\,m$. The distance from the front axle to the rear axle $L = 3.0\,m$. The height of top view point in CARLA is 50 m. The important parameters of the NNs are shown in Table 3, and the computational complexity, e.g., FLOPs and model's parameter sizes are shown in Tables 4 and 5.

Offline training

The ACOPFM, GEP, and l_1 exact penalty function based method (denoted by EPF) are trained for comparison. In the key iteration step of the GEP [30] 'if i mod m: $\rho \leftarrow c\rho$', the parameter settings are shown in Table 3. Five different runs are conducted with different random seeds on a desktop computer with a 3.70 GHz-10 core Intel Xeon W-2255 CPU, with evaluations every 100 iterations. The GEP and the EPF are trained using four different sets of initial penalty parameters ρ_0, and are compared with the ACOPFM. The numerical results are reported in Fig. 4.

As seen in Fig. 4a, from the trend of J_{actor}, compared with the GEP and the EPF, we can see that ACOPFM converges faster, and is more stable; in other words, ACOPFM is able to find the optimal strategy more efficiently, which means that the trajectory tracking performance of ACOPFM is better than that of the GEP and the EPF. Figure 4b shows the variation of the objective penalty parameter M; compared with the infinite growth of the penalty parameter ρ of the GEP and EPF as shown in Fig. 4c, it converges faster to a finite value that is roughly equal to the parameterized J_{actor}. Figure 4d shows the variation of J_p, and it can be seen that ACOPFM converges to a small number quite steadily, whereas the GEP

Table 2 The starting and ending points' coordinates

Reference path number	Point	Coordinates
1	Start	carla.Location(x=106.3154, y=237.56, z=0.3)
	End	carla.Location(x=599.10, y=237.73, z=0.3)
2	Start	carla.Location(x=106.3154, y=241.06, z=0.3)
	End	carla.Location(x=599.10, y=241.06, z=0.3)
3	Start	carla.Location(x=106.3154, y=244.56, z=0.3)
	End	carla.Location(x=599.10, y=244.73, z=0.3)
4	Start	carla.Location(x=106.3154, y=248.06, z=0.3)
	End	carla.Location(x=599.10, y=248.06, z=0.3)
5	Start	carla.Location(x=106.3154, y=251.56, z=0.3)
	End	carla.Location(x=599.10, y=251.73, z=0.3)

Table 3 Important hyperparameter settings

Hyperparameter	Value
Optimizer	Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$)
Approximation function	MLP
Number of hidden layers	2
Number of hidden units	256
Nonlinearity of hidden layer	ELU
Replay buffer size $\mathcal{B}$	1e4
Batch size	1024
Policy learning rate (actor)	1e-6
Value learning rate (critic)	1e-6
The penalty amplifier c of the GEP and the EPF	1.1
The update interval m of the GEP and the EPF	150
The initial Parameter a_0 of ACOPFM	–1
Total iteration	20,000

Table 4 Computational complexity of the policy network

Layer (type)	Output shape	Param #
Linear-1	[1024, 1, 256]	8704
ELU-2	[1024, 1, 256]	0
Linear-3	[1024, 1, 256]	65,792
ELU-4	[1024, 1, 256]	0
Linear-5	[1024, 1, 2]	514
Total params: 75010	Input size (MB): 0.13	
Trainable params: 75010	Forward/backward pass size (MB): 8.02	
Non-trainable params: 0	Params size (MB): 0.29	
Flops: 76283904.0	Estimated total size (MB): 8.43	

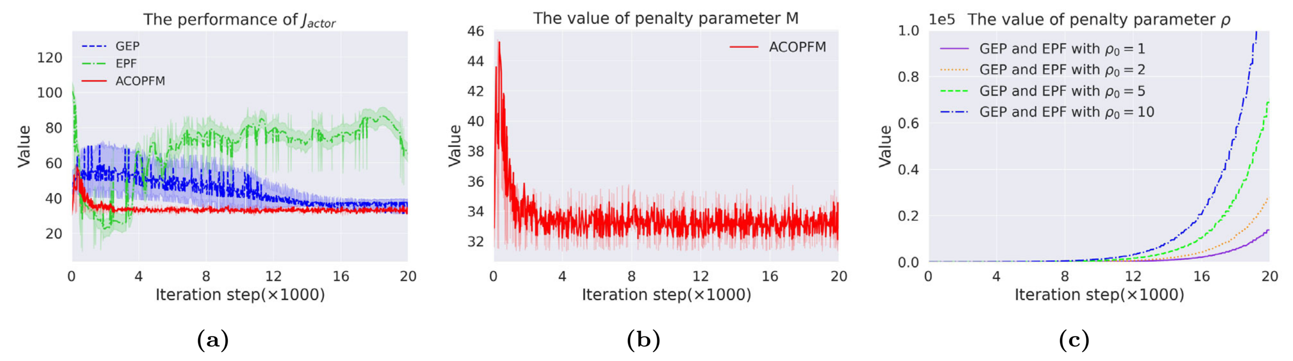
gives a diverging and oscillating trend, and the EPF gives an oscillating result. The numerical performance of the value network of the three methods are shown in Fig. 4e. It can be seen that the ACOPFM converges very fast, whereas the GEP and the EPF diverge. Figure 4f demonstrates the variation of $J_{penalty}$, which indicates the degree of constraint violations. It can be seen that the ACOPFM performs much better than the GEP and the EPF. Moreover, we take the average violation degree [65] $\bar{J}_{penalty}$ of $J_{penalty}$ to measure the safety performance:

$$\bar{J}_{penalty} = \mathbb{E}_{0|k}\left\{\mathbb{E}_{s_{0|t}}\left\{\sum_{i=0}^{T-1}\phi_i(\theta)\right\}\right\}. \tag{13}$$

Overall, the average violation degree of ACOPFM is lower than the average value of the GEP and the EPF as shown

Table 5 Computational complexity of the value network

Layer (type)	Output Shape	Parameter #
Linear-1	[1024, 1, 256]	8,704
ELU-2	[1024, 1, 256]	0
Linear-3	[1024, 1, 256]	65,792
ELU-4	[1024, 1, 256]	0
Linear-5	[1024, 1, 1]	257
Total params: 74573	Input size (MB): 0.13	
Trainable params: 74573	Forward/backward pass size (MB): 8.01	
Non-trainable params: 0	Params size (MB): 0.29	
Flops: 76021760.0	Estimated Total Size (MB): 8.43	



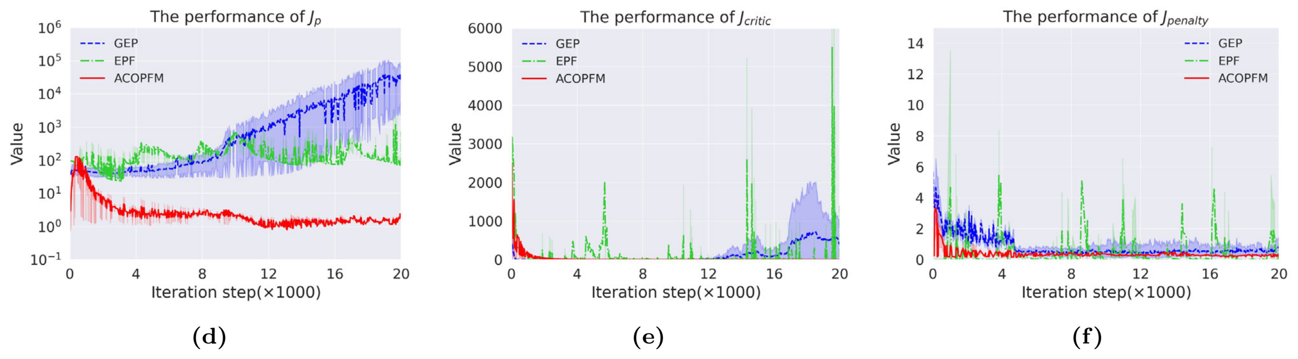


Fig. 4 Comparative experimental results of the ACOPFM with the GEP, the EPF in a multi-lane overtaking simulation scenario. The solid lines represent the mean value over different episodes. The shaded regions represent the 95% confidence interval. **a** The variation of J_{actor}. **b** The objective penalty parameter M. **c** The penalty parameter ρ of the GEP and the EPF. **d** The variation of J_p. **e** The variation of J_{critic}. **f** The variation of $J_{penalty}$

in Table 6, which indicates that the safety performance of ACOPFM is better than that of the GEP and the EPF in general.

Online testing

We deploy the model trained by the ACOPFM to the ego vehicle and test it in a Town06 multi-lane scenario. The red thick line segment is the optimized trajectory obtained by the ACOPFM, which is formed by connecting waypoints within the prediction horizon ($T = 30$) at each simulation moment. The green solid line is the reference path, generated by the A^* algorithm in the CARLA simulator after considering the road static information, and there are 5 reference paths corresponding to the multi-lane scenario.

1) Performance in the environment without surrounding vehicles

In such environment, the ego vehicle trajectory should be well maintained in the reference lane. In the SUMO simulator we set the vehicle generation frequency ***vehsPerHour*** to

Table 6 Average violation degree $\bar{J}_{penalty}$ of the ACOPFM, GEP and EPF

Method	$k = 5000$	$k = 10000$	$k = 15000$	$k = 20000$
ACOPFM	**0.076573**	**0.038286**	**0.025524**	**0.019143**
The GEP with $\rho_0 = 1$	0.113201	0.056600	0.037734	0.028300
The GEP with $\rho_0 = 2$	0.128813	0.064406	0.042938	0.032203
The GEP with $\rho_0 = 5$	0.151513	0.075757	0.050504	0.037878
The GEP with $\rho_0 = 10$	0.256800	0.128400	0.085600	0.064200
Average value of the GEP	**0.162582**	**0.081291**	**0.054194**	**0.04065**
EPF with $\rho_0 = 1$	0.098345	0.070590	0.047060	0.035295
EPF with $\rho_0 = 2$	0.141179	0.201405	0.134270	0.100702
EPF with $\rho_0 = 5$	0.402809	0.203583	0.135722	0.101792
EPF with $\rho_0 = 10$	0.624790	0.312395	0.208263	0.156197
Average value of EPF	**0.316781**	**0.196993**	**0.131329**	**0.098450**

The bold values are used for comparison

0 vehicles per hour. The dynamic breakdown of the whole process of the simulation test is shown in Fig. 5, where the ego vehicle is initialized in the lane 2, with the goal of reaching the end of the line in a safe and traffic rules compliance manner.

During the whole test process, the changes of the indicators of the ego vehicle are shown in Fig. 6, and it can be seen that the ego vehicle almost keeps a constant speed after accelerating for a period of time, and the front-wheel angle, the steering wheel angle and the heading angle parameters are basically kept near 0, which indicates that the trajectory tracking performance of the ego vehicle is good.

2) Performance in the environment with surrounding vehicles

In this environment, the challenging task is to safely and stably complete the overtaking control of the ego vehicle. We adopt a multistep safety shield [30] after the output of the policy to enhance safety during the online implementation of the trained model. In the SUMO simulator we set the vehicle generation frequency ***vehsPerHour*** to 5000 vehicles per hour. The dynamic breakdown of the whole process of the simulation test is shown in Figs. 7 and 8.

A collision can occur when there is a vehicle A in front of the ego vehicle. The ego vehicle will turn to lane 2, as shown in Fig. 7b, and chooses lane 2 as the path to be tracked to complete the first overtaking, as shown in Fig. 7c and d.

After driving stably on lane 2 for a period of time, the sensor detects vehicle B which is directly ahead of the ego vehicle, as shown Fig. 8b, and the ego vehicle will perform a second lane change for overtaking. The ego vehicle chooses to change lane to enter the lane 1 to the left and overtake the vehicle B by acceleration, as shown in Fig. 8c. After the second overtaking, the ego vehicle ultimately arrives at the end of the lane safely and stably, as shown in Fig. 8c and d.

During the whole test process, the changes of the indicators of the ego vehicle are shown in Fig. 9. Around simulation step 170 and step 530, it is evident that there are clear fluctuations in the indicators, since the first and second overtaking maneuvers entail the changes of the acceleration and the front-wheel angle, which are reflected in Fig. 9a to Fig. 9f). Note that the first and second overtaking maneuvers involve the right-turn control and left-turn control of the steering wheel, which are implemented by changing the angle of the steering wheel (Fig. 9c). The front-wheel angle (Fig. 9d) and the heading angle (Fig. 9e) have abrupt changes around simulation step 170 and step 530.

Conclusion and future work

In this paper, we have proposed a model-based RL algorithm, the ACOPFM, to avoid the ill-conditioning issue faced by the GEP during model training and effectively deal with the large-scale nonlinear state-wise constraints. A planning and control scheme based on the IDC framework is adopted, and a MPC-based model of the trajectory tracking problem is established. Subsequently, the model is converted into an unconstrained optimization problem using the objective penalty function method and it is parameterized based on the actor-critic framework. The loss function guiding the update of the policy network is employed in the form of objective penalty function. The ACOPFM can find the optimal control strategy faster and stabler in comparison with the GEP by alternately performing gradient descent and adaptively adjusting the penalty parameter. The simulation results show that under multi-lane scenarios, the ACOPFM performs well in trajectory tracking tasks and is able to complete the overtaking maneuvers successfully.

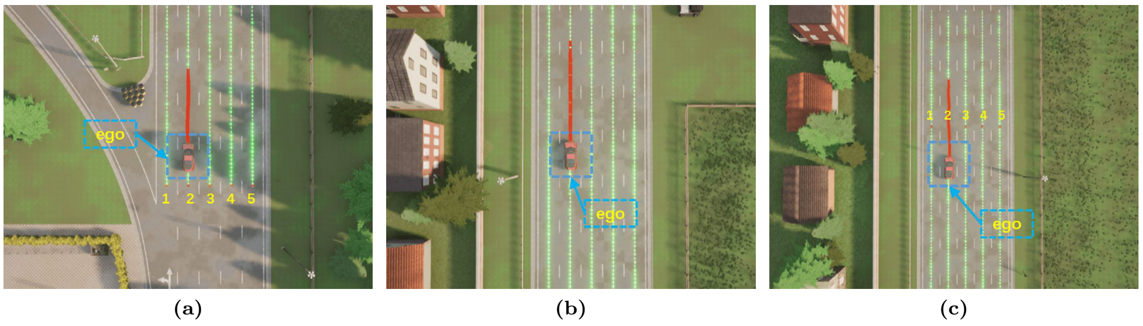


Fig. 5 Visualization of the trajectory tracking simulation process without interactions with other vehicles. **a** The ego vehicle starts in lane 2. **b** The ego vehicle is in motion. **c** The ego vehicle is approaching the end (marked by red dot) smoothly

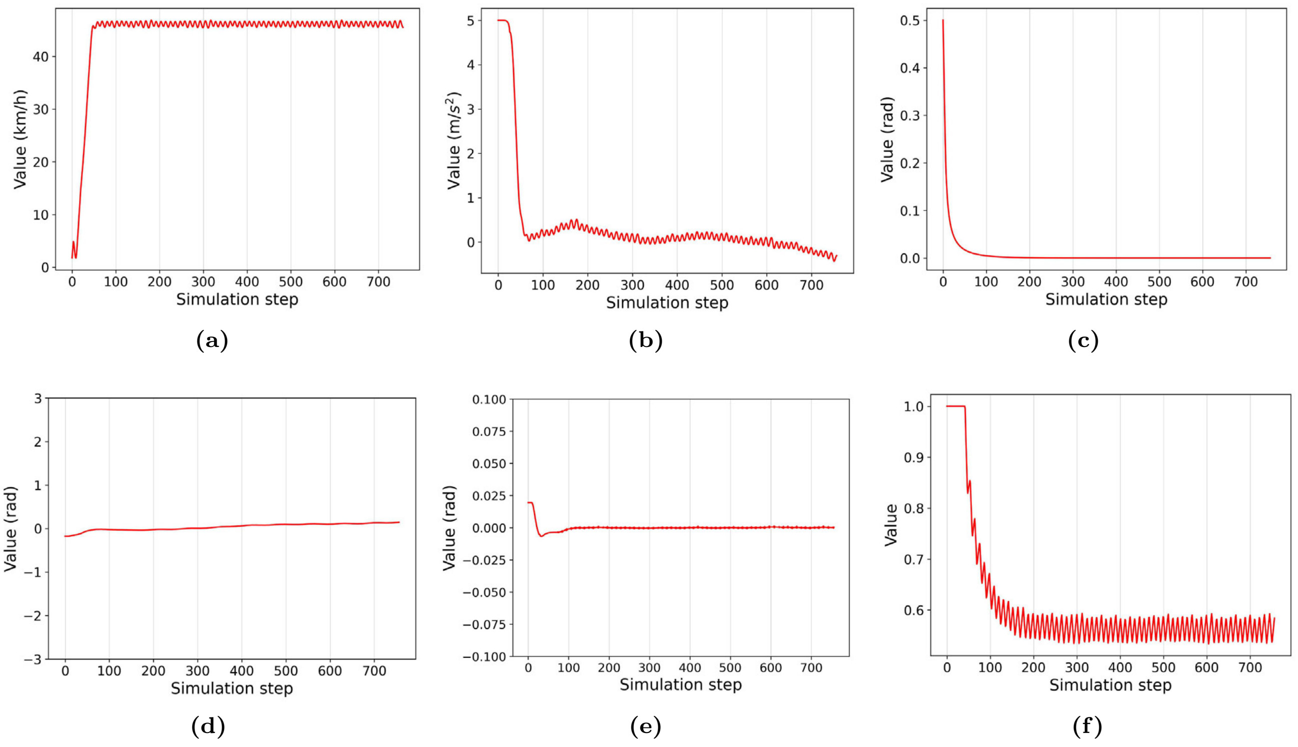


Fig. 6 Results of the indicators in the environment without interactions with other vehicles. **a** The variation of the speed. **b** The variation of the acceleration. **c** The variation of the steering wheel angle. **d** The variation of the front-wheel angle. **e** The variation of the heading angle. **f** The variation of the throttle parameter

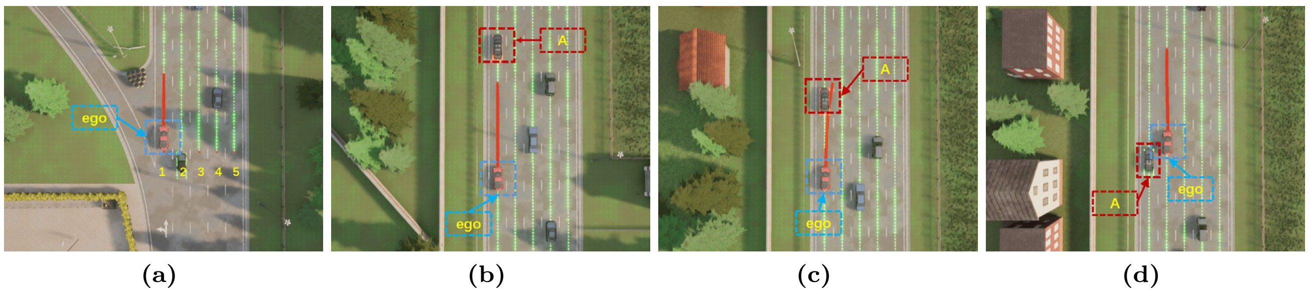


Fig. 7 Visualization of the ego vehicle's maneuvers during the first overtaking. **a** The ego vehicle starts in the lane 1. **b** The ego vehicle detects vehicle *A* in the right front of it through sensors in the CARLA simulator and prepares to steer itself enter lane 2 to the right. **c** The ego vehicle interacts with vehicle *A* safely and stably. **d** The ego vehicle completes the first lane change to overtake and travels on lane 2

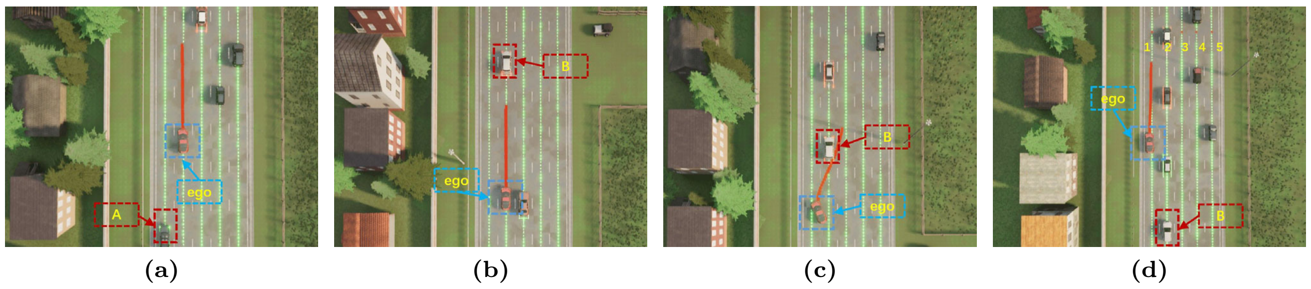


Fig. 8 Visualization of the ego vehicle's maneuvers during the second overtaking. **a** The ego vehicle travels on lane 2 for some time. **b** The ego vehicle detects vehicle *B* in the right front of it via sensors in the CARLA simulator and prepares to steer itself to enter lane 1 on the left. **c** The ego vehicle interacts with vehicle *B* safely and stably. **d** The ego vehicle completes the second lane change to overtake and travels on lane 1 to reach the end (marked with red dot) successfully

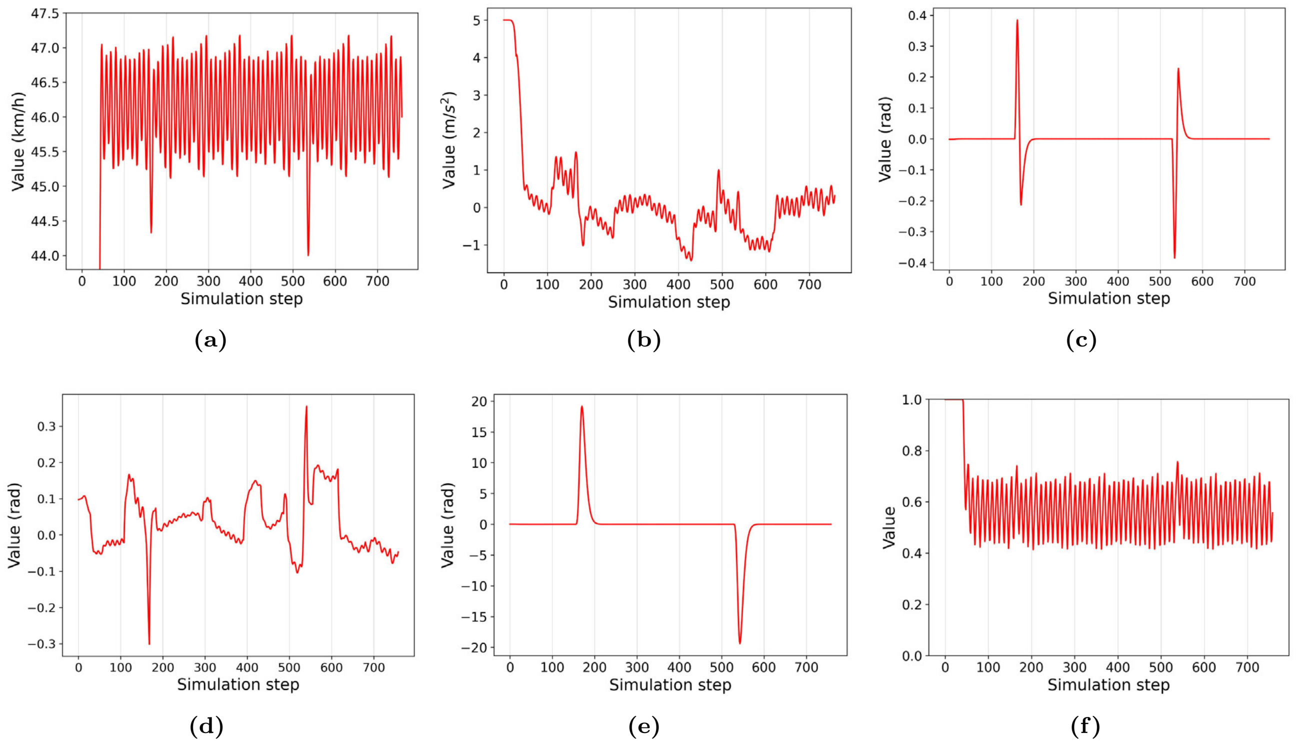


Fig. 9 Results of indicators in the environment with surrounding vehicles. **a** Variation of the speed. **b** Variation of the acceleration. **c** Variation of the steering wheel angle. **d** Variation of the front-wheel angle. **e** Variation of the heading angle. **f** Variation of the throttle parameter

As the ACOPFM converts the original problem into an unconstrained optimization problem by imposing penalties for constraint violations as the GEP, there is no 100% guarantee of safety. Our next research is to improve the multistep safety shields [30] to ensure safety performance.
Note that the state transfer equation we employed in the present work is relatively simple and there could be errors in real applications. There are some environmental factors such as wind speed, ambient temperature, and ground slope that are not considered in the simulation process. Moreover, some scenes (e.g., some control behaviors of the ego vehicle and other vehicles) are not encountered during model training, which can cause long-tail problems and makes the real applications of the method less secure. However, by refining the model and extending the training process, these problems can be overcome to a large extent.

Acknowledgements The authors would like to thank the anonymous referees for their very helpful comments and insightful suggestions that have helped improve this paper greatly. This work was supported by the Chongqing Technological Innovation and Applications Development Special Program (Project No. cstc2021jscx-jbgsX0001), the Scientific and Technological Research Program of the Chongqing Municipal Education Commission (Project No. KJZD-K202114801), the Innovation and Development Joint Program of the Chongqing Natural Science

Foundation (Project No. 2022NSCQ-LZX0301) and the National Natural Science Foundation of China (Project No. 11991024).

Data Availability The data associated with this paper is available from the corresponding author upon request.

Declarations

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

References

1. Badue Claudine, Guidolini Rânik, Carneiro Raphael Vivacqua, Azevedo Pedro, Cardoso Vinicius B, Forechi Avelino, Jesus Luan, Berriel Rodrigo, Paixao Thiago M, Mutz Filipe, et al (2021) Self-driving cars: A survey. *Expert Systems with Applications*, 165:113816
2. González D, Pérez J, Milanés V, Nashashibi F (2015) A review of motion planning techniques for automated vehicles. IEEE Trans Intell Transp Syst 17(4):1135–1145
3. Huang Z, Li H, Li W, Liu J, Huang C, Yang Z, Fang W (2021) A new trajectory tracking algorithm for autonomous vehicles based on model predictive control. Sensors 21(21):7165
4. Chatzikomis C, Sorniotti A, Gruber P, Zanchetta M, Willans D, Balcombe B (2018) Comparison of path tracking and torque-vectoring controllers for autonomous electric vehicles. IEEE Transactions on Intelligent Vehicles 3(4):559–570
5. Li L, Li J, Zhang S (2021) Review article: State-of-the-art trajectory tracking of autonomous vehicles. Mechanical Sciences 12(1):419–432
6. Shtessel Yuri, Edwards Christopher, Fridman Leonid, Levant Arie, et al (2014) *Sliding mode control and observation*, volume 10. Springer
7. Karl Johan Åström and Tore Hägglund (2001) The future of pid control. Control Eng Pract 9(11):1163–1175
8. Grüne Lars, Pannek Jürgen, Grüne Lars, Pannek, Jürgen (2017) *Nonlinear model predictive control*. Springer
9. Liu J-K, Sun F-C (2007) Research and development on theory and algorithms of sliding mode control. Kongzhi Lilun yu Yingyong/Control Theory & Applications 23(3):407–418
10. Kachroo P, Tomizuka M (1996) Chattering reduction and error convergence in the sliding-mode control of a class of nonlinear systems. IEEE Trans Autom Control 41(7):1063–1068
11. Huang B, Yang Q (2019) Double-loop sliding mode controller with a novel switching term for the trajectory tracking of work-class rovs. Ocean Eng 178:80–94
12. Elmokadem T, Zribi M, Youcef-Toumi K (2016) Trajectory tracking sliding mode control of underactuated auvs. Nonlinear Dyn 84:1079–1091
13. Labbadi M, Cherkaoui M (2019) Robust adaptive backstepping fast terminal sliding mode controller for uncertain quadrotor uav. Aerosp Sci Technol 93:105306
14. Ge Q, Sun Q, Li SE, Zheng S, Wu W, Chen X (2021) Numerically stable dynamic bicycle model for discrete-time control. In: 2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops), pp 128–134. IEEE
15. Mohan Tiwari Pyare, Janardhanan S, un Nabi Mashuq (2015) Rigid spacecraft attitude control using adaptive non-singular fast terminal sliding mode. Journal of Control, Automation and Electrical Systems 26:115–124
16. Hassani H, Mansouri A, Ahaitouf A (2021) Robust autonomous flight for quadrotor uav based on adaptive nonsingular fast terminal sliding mode control. Int J Dyn Control 9:619–635
17. Rupp Astrid, Stolz Michael (2017) Survey on control schemes for automated driving on highways. In *Automated driving*, pages 43–69. Springer
18. Nie L, Guan J, Chihua L, Zheng H, Yin Z (2018) Longitudinal speed control of autonomous vehicle based on a self-adaptive pid of radial basis function neural network. IET Intel Transp Syst 12(6):485–494
19. Howard Thomas M, Alonzo K (2007) Optimal rough terrain trajectory generation for wheeled mobile robots. Int J Robot Res 26(2):141–166
20. Li S, Li K, Rajamani R, Wang J (2010) Model predictive multi-objective vehicular adaptive cruise control. IEEE Trans Control Syst Technol 19(3):556–566
21. Sutton Richard S, Barto Andrew G (2018) Reinforcement learning: an introduction. MIT press
22. Pal Constantin-Valentin, Leon Florin (2020) Brief survey of model-based reinforcement learning techniques. In *2020 24th International Conference on System Theory, Control and Computing (ICSTCC)*, pages 92–97. IEEE
23. Bellman R (1966) Dynamic programming. Science 153(3731):34–37
24. Xingwei Z, Bo Tao L, Qian HD (2020) Model-based actor-critic learning for optimal tracking control of robots with input saturation. IEEE Trans Industr Electron 68(6):5046–5056
25. Yu Lingli, Shao Xuanya, Yan Xiaoxin (2017) Autonomous overtaking decision making of driverless bus based on deep q-learning method. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2267–2272. IEEE
26. Tang X, Huang B, Liu T, Lin X (2022) Highway decision-making and motion planning for autonomous driving via soft actor-critic. IEEE Trans Veh Technol 71(5):4706–4717
27. Zanon M, Gros S (2020) Safe reinforcement learning using robust mpc. IEEE Trans Autom Control 66(8):3638–3652
28. Gros S, Zanon M (2019) Data-driven economic nmpc using reinforcement learning. IEEE Trans Autom Control 65(2):636–648
29. Gros Sébastien, Zanon Mario (2021) Reinforcement learning based on MPC and the stochastic policy gradient method. In *2021 American Control Conference (ACC)*, pages 1947–1952. IEEE
30. Yang G, Yangang R, Qi S, Eben LS, Haitong M, Jingliang D, Yifan D, Bo C (2022) Integrated decision and control: toward interpretable and computationally efficient driving intelligence. IEEE transactions on cybernetics 53(2):859–873
31. Brian P, Michal Č, Zheng YS, Dmitry Y, Emilio F (2016) A survey of motion planning and control techniques for self-driving urban vehicles. IEEE Transactions on intelligent vehicles 1(1):33–55
32. Ravi KB, Ibrahim S, Victor T, Patrick M, Al Sallab Ahmad A, Senthil Y, Patrick P (2021) Deep reinforcement learning for autonomous driving: A survey. IEEE Trans Intell Transp Syst 23(6):4909–4926

33. Fletcher R_ (1981) Practical methods of optimization: Vol. 2: Constrained optimization. *JOHN WILEY & SONS, INC., ONE WILEY DR., SOMERSET, N. J. 08873, 1981, 224*
34. Charalambous Christakis (1980) A method to overcome the ill-conditioning problem of differentiable penalty functions. *Operations Research*, 28(3-part-ii):650–667
35. Fletcher Roger (1983) Penalty functions. *Mathematical Programming The State of the Art*, pages 87–114
36. Dussault J-P (1995) Numerical stability and efficiency of penalty algorithms. SIAM J Numer Anal 32(1):296–317
37. Saarinen S, Bramley R, Cybenko G (1993) Ill-conditioning in neural network training problems. SIAM J Sci Comput 14(3):693–714
38. Zhang Yongke, Zhang Yongjun, Ye Wei (1995) Local-sparse connection multilayer networks. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 3, pages 1254–1257. IEEE
39. Der Smagt Patrick Van, Hirzinger Gerd (2002) Solving the ill-conditioning in neural network learning. In *Neural networks: tricks of the trade*, pages 193–206. Springer
40. Byrd Richard H, Gabriel L-C, Jorge N (2012) A line search exact penalty method using steering rules. Math Program 133(1):39–73
41. Rheinboldt Werner C (1976) On measures of ill-conditioning for nonlinear equations. Math Comput 30(133):104–111
42. Peters G, Wilkinson James H (1979) Inverse iteration, ill-conditioned equations and newton's method. SIAM Rev 21(3):339–360
43. Peter KM, Chua Leon O (1988) Neural networks for nonlinear programming. IEEE Transactions on Circuits and Systems 35(5):554–562
44. Jie L, Gupte A, Huang Y (2018) A mean-risk mixed integer nonlinear program for transportation network protection. Eur J Oper Res 265(1):277–289
45. Nocedal Jorge, Wright Stephen J (2006) *Numerical Optimization*, 2nd edition. Springer
46. Luenberger David G, Ye Yinyu (2021) *Linear and Nonlinear Programming*, 5th edition. Springer Nature Switzerland AG
47. Murray W (1967) Ill-conditioning in barrier and penalty functions arising in constrained nonlinear programming. In *Proceedings of the Sixth International Symposium on Mathematical Programming*
48. Zangwill Willard I (1967) Non-linear programming via penalty functions. Manage Sci 13(5):344–358
49. Coleman Thomas F, Conn Andrew R (1980) Second-order conditions for an exact penalty function. Math Program 19(1):178–185
50. Körner F (1990) On the numerical realization of the exact penalty method for quadratic programming algorithms. Eur J Oper Res 46(3):404–408
51. Mongeau M, Sartenaer A (1995) Automatic decrease of the penalty parameter in exact penalty function methods. Eur J Oper Res 83(3):686–699
52. Morrison David D (1968) Optimization by least squares. SIAM J Numer Anal 5(1):83–88
53. Meng Z, Qiying H, Dang C, Yang X (2004) An objective penalty function method for nonlinear programming. Appl Math Lett 17(6):683–689
54. Meng Z, Qiying H, Dang C (2009) A penalty function algorithm with objective parameters for nonlinear mathematical programming. Journal of Industrial & Management Optimization 5(3):585
55. Meng Z, Dang C, Jiang M, Xinsheng X, Shen R (2013) Exactness and algorithm of an objective penalty function. J Global Optim 56(2):691–711
56. Min J, Meng Z, Zhou G, Shen R (2021) On the smoothing of the norm objective penalty function for two-cardinality sparse constrained optimization problems. Neurocomputing 458:559–565
57. Anil A, Humberto G, Shankar SS, Claire T (2013) Provably safe and robust learning-based model predictive control. Automatica 49(5):1216–1226
58. Koller Torsten, Berkenkamp Felix, Turchetta Matteo, Krause Andreas (2018) Learning-based model predictive control for safe exploration. In *2018 IEEE conference on decision and control (CDC)*, pages 6059–6066. IEEE
59. Zanon Mario, Gros Sébastien, Bemporad Alberto (2019) Practical reinforcement learning of stabilizing economic mpc. In *2019 18th European Control Conference (ECC)*, pages 2258–2263. IEEE
60. Arroyo J, Manna C, Spiessens F, Helsen L (2022) Reinforced model predictive control (rl-mpc) for building energy management. Appl Energy 309:118346
61. Garcia Carlos E, Prett David M, Manfred M (1989) Model predictive control: Theory and practice-a survey. Automatica 25(3):335–348
62. Karg B, Lucia S (2020) Efficient representation and approximation of model predictive control laws via deep learning. IEEE Transactions on Cybernetics 50(9):3866–3878
63. Chen Jianyu, Li Shengbo Eben, Tomizuka Masayoshi (2021) Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*
64. Ren Yangang, Duan Jingliang, Li Shengbo Eben, Guan Yang, Sun Qi (2020) Improving generalization of reinforcement learning with minimax distributional soft actor-critic. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE
65. Ma Haitong, Chen Jianyu, Eben Shengbo, Lin Ziyu, Guan Yang, Ren Yangang, Zheng Sifa (2021) Model-based constrained reinforcement learning using generalized control barrier function. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4552–4559. IEEE