Integrated Decision and Control: Toward Interpretable and Computationally Efficient Driving Intelligence

Yang Guan, Yangang Ren, Qi Sun, Shengbo Eben Li, *Senior Member, IEEE*, Haitong Ma, *Graduate Student Member, IEEE*, Jingliang Duan, Yifan Dai, and Bo Cheng

***Abstract*—Decision and control are core functionalities of high-level automated vehicles. Current mainstream methods, such as functional decomposition and end-to-end reinforcement learning (RL), suffer high time complexity or poor interpretability and adaptability on real-world autonomous driving tasks. In this article, we present an interpretable and computationally efficient framework called integrated decision and control (IDC) for automated vehicles, which decomposes the driving task into static path planning and dynamic optimal tracking that are structured hierarchically. First, the static path planning generates several candidate paths only considering static traffic elements. Then, the dynamic optimal tracking is designed to track the optimal path while considering the dynamic obstacles. To that end, we formulate a constrained optimal control problem (OCP) for each candidate path, optimize them separately, and follow the one with the best tracking performance. To unload the heavy online computation, we propose a model-based RL algorithm that can be served as an approximate-constrained OCP solver. Specifically, the OCPs for all paths are considered together to construct a single complete RL problem and then solved offline in the form of value and policy networks for real-time online path selecting and tracking, respectively. We verify our framework in both simulations and the real world. Results show that compared with baseline methods, IDC has an order of magnitude higher online computing efficiency, as well as better driving performance, including traffic efficiency and safety. In addition, it yields great interpretability and adaptability among different driving scenarios and tasks.**

***Index Terms*—Automated vehicle, decision and control, model based, reinforcement learning (RL).**

Manuscript received 29 April 2021; revised 3 October 2021 and 22 January 2022; accepted 25 March 2022. Date of publication 19 April 2022; date of current version 13 January 2023. This work was supported in part by the International Science and Technology Cooperation Program of China under Grant 2019YFE0100200; in part by NSF China, under Grant U20A20334 and Grant 52072213; and in part by the Tsinghua University-Toyota Joint Research Center for AI Technology of Automated Vehicle. This article was recommended by Associate Editor X. Qu. *(Corresponding author: Shengbo Eben Li.)*

Yang Guan, Yangang Ren, Qi Sun, Shengbo Eben Li, Haitong Ma, Jingliang Duan, and Bo Cheng are with the School of Vehicle and Mobility, Tsinghua University, Beijing 100084, China (e-mail: lisb04@gmail.com).

Yifan Dai is with the Suzhou Automotive Research Institute, Tsinghua University, Suzhou 215200, China.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCYB.2022.3163816.

Digital Object Identifier 10.1109/TCYB.2022.3163816

I. Introduction

INTELLIGENCE of automobile technology and driving assistance system has great potential to improve safety, reduce fuel consumption, and enhance traffic efficiency, which will completely change road transportation. Decision and control are indispensable for high-level autonomous driving, which is in charge of computing the expected instructions of steering and acceleration relying on the environment perception results. It is generally believed that there are two technical routes for the decision and control of automated vehicles: 1) decomposed scheme and 2) end-to-end scheme.

The decomposed scheme splits the decision and control functionality into several serial submodules, such as scene understanding, prediction, behavior selection, trajectory planning, and control [1]. Scene understanding is to dig more relevant information from perception to help the following processes [2]. Prediction is to predict the future trajectory of traffic participants to determine the feasible region in the future [3]. It is further decomposed into behavior recognition [4], [5] and trajectory prediction [6], [7]. Since prediction algorithms usually work on each surrounding vehicle, the more vehicles, the more computation is needed. Behavior selection then chooses a high-level driving behavior relying on an expert system in which integrates many designed rules. Typical methods include finite-state machine [8] and decision tree [9]. Based on the selected behavior, a collision-free spacetime curve satisfying vehicle dynamics is calculated by the trajectory planning submodule, according to predicted trajectories and road constraints. Two main categories of the planning algorithms include optimization-based and search-based. Optimization-based methods formulate the planning problem into an optimization problem, where specific aspects of trajectory are optimized and constraints are considered [10]–[13]. However, it suffers from high computational complexity for large-scale nonlinear and nonconvex problems. The search-based methods represented by A* and rapidly exploring random tree are more efficient [14]–[16]. Still, they usually lead to low-resolution paths and can barely consider dynamic obstacles. Xin *et al.* [17] proposed a combination of the optimization-based and search-based methods, where a trajectory is searched in the spacetime by A* and then smoothed by model predictive control (MPC), yielding the best

performance in terms of planning time and comfort. Finally, the controller is used to follow the planned trajectory and calculate the expected controls [18], [19]. The decomposed scheme requires a large amount of human design, but it is still hard to cover all possible driving scenarios due to the long tail effect. Besides, it cannot guarantee real-time performance because it is time consuming to complete all the works serially in a limited time for industrial computers.

The end-to-end scheme computes the expected instructions directly from inputs given by the perception module using a policy usually carried out by a deep neural network (NN). Reinforcement learning (RL) methods do not rely on labeled driving data but learn by trial and error in real world, or a high fidelity simulator [20]–[22]. Early RL applications on autonomous driving mainly focus on learning a single driving behavior, e.g., lane keeping [23], lane changing [24], or overtaking [25]. They usually employ deep Q-networks [26], [27] or deep deterministic policy gradient method [28] to learn policy in discrete or continuous domain. Besides, they own different reward functions for their respective goals. Recently, RL has been applied in certain driving scenarios. Duan *et al.* [29] realized decision making under a virtual two-lane highway using hierarchical RL, which designs complicated reward functions for its high-level maneuver selection and three low-level maneuvers, respectively. Guan *et al.* [30] achieved centralized control in a four-leg single-lane intersection with sparse rewards, in which only eight cars are considered. Chen *et al.* [31] designed a bird-view representation and used visual encoding to capture the low-dimensional latent states, solving the driving task in a roundabout scenario with dense surrounding vehicles in a high-definition driving simulator. However, they reported limited safety performance and poor learning efficiency. Current RL methods mostly work on a specific task, in which a set of complicated reward functions is required to offer guidance for policy optimization, such as distance traveled toward a destination, collisions with other road users or scene objects, and maintaining comfort and stability while avoiding extreme acceleration, braking, or steering. It is nontrivial and needs a lot of human effort to tune, causing poor adaptability among driving scenarios and tasks. Besides, the outcome of the policy is hard to interpret, which makes it barely used in real autonomous driving tasks. Moreover, they cannot deal with safety constraints explicitly and suffer from low convergence speed.

This article proposes an integrated decision and control (IDC) framework for automated vehicles, which has great interpretability and online computing efficiency and is applicable in different driving scenarios and tasks. The contributions emphasize in three parts which are described as follows.

1) We proposed an IDC framework for automated vehicles, which decomposes driving tasks into static path planning and dynamic optimal tracking hierarchically. The high-level static path planning is used to generate multiple paths only considering static constraints, such as road topology and traffic lights. The low-level dynamic optimal tracking is used to select the optimal path and track it considering dynamic obstacles, wherein a finite-horizon-constrained optimal control problem (OCP) is constructed and optimized for each candidate path. The optimal path is selected as the one with the lowest optimal cost function. Compared to existing functional decomposition methods, the IDC is much more efficient in computation because it unloads the heavy online optimizations of the constrained OCPs in offline using RL. As a result, two NNs called value and policy are solved for online path selecting and tracking, respectively. On the other hand, compared to the end-to-end scheme, it is interpretable because the value and policy NNs are the approximation for the optimal cost and action of the constrained OCP, which makes the result more explicit to understand and predict. Moreover, instead of designing and tuning case by case, the formulated OCP in the IDC is task independent with tracking objective and safety constraints, making it applicable among a variety of scenarios and tasks.
2) We develop a model-based RL algorithm called the generalized exterior point method (GEP) to solve OCP with large-scale statewise constraints approximately. The GEP is an extension of the exterior point method in the optimization domain to the field of NN, where it first constructs an extensive problem involving all the candidate paths and transforms it into an unconstrained problem with a penalty on safety violations. Then, the approximate feasible optimal control policy is obtained by alternatively performing gradient descent and enlarging the penalty. Existing methods are mainly proposed to solve constrained Markov decision processes (MDPs), whose format is only with a single constraint. In contrast, the GEP can explicitly deal with problems with large-scale statewise constraints and efficiently update NNs with the guidance of the model. Besides, it relaxes the convergence condition of the exterior point method so that the global minimum in each gradient descent step is not required. To the best of our knowledge, GEP is the first model-based solver for OCPs with large-scale statewise constraints that NNs parameterize.
3) We evaluate the proposed method extensively in both simulations and in a real-world road to verify online computing efficiency, safety, task adaptation, etc. The results show the method's potential to be applied in real-world autonomous driving tasks.

II. Integrated Decision and Control Framework

This section introduces the framework of IDC. As shown in Fig. 1, the framework consists of two layers: 1) static path planning and 2) dynamic optimal tracking.

Unlike existing schemes, the upper layer aims to generate multiple candidate paths only considering static information, such as road structure, speed limit, traffic signs, and lights. Note that these paths will not include time information. Each candidate path is attached with an expected velocity determined by rules from human experience. Similar to existing trajectory planning methods, there is no optimality guarantee with the path generation. However, the time complexity of the path selection in our approach remains unchanged

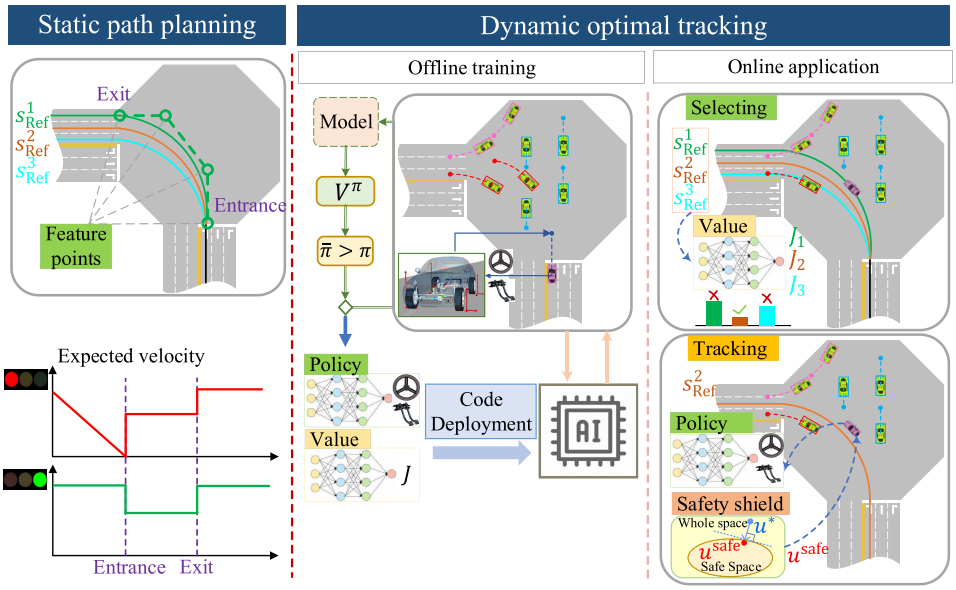


Fig. 1. Illustration of the IDC framework.

with the growth of the candidate path set. Therefore, we can increase the number of reference paths according to our demand until the required performance is met. For example, at an intersection, the candidate path can be as many as the exit lanes; on the highway, the candidate path can include the centerline of the current and adjacent lanes, etc.

The lower layer further considers the candidate paths and the dynamic information, such as surrounding vehicles, pedestrians, and bicycles. A constrained OCP is formulated and optimized for each candidate path to choose the optimal path and find the control command. The objective function minimizes the tracking error within a finite horizon, and the constraints characterize safety requirements. The optimal path is chosen with the lowest optimal cost function in each time step and then tracked. The core of our method is to substitute all the expensive online optimizations with feedforward of two NNs trained offline by RL. Precisely, we first formulate a complete RL problem considering all the candidate paths. And then, we develop a model-based RL algorithm to solve this problem to obtain a policy NN called actor capable of tracking different shapes of paths while maintaining the ability to avoid collisions. Meanwhile, a value NN called critic is learned to approximate the optimal cost of following separate paths for online path selection.

The advantages of the IDC framework are summarized in three points. First, it has high online computing efficiency. The upper layer can be quite efficient because it involves only static information. It even allows embedding key path parameters into the electronic map. On the other hand, the lower layer utilizes two trained networks to compute optimal path and control command, which is also time saving due to the fast propagation of NNs. Second, we can quickly transfer it among different driving scenarios without much human design. As the upper layer only uses static information, the multiple paths can be easily generated by the road topology of various scenes, such as intersections, roundabouts, and ramps. Besides, the lower layer always formulates a similar tracking problem with safety constraints no matter what the task is. All can be solved by the developed model-based solver, saving time to design separate reward functions for different tasks. Third, the IDC framework has great interpretability. Interpretability is defined as the degree to which a human can understand the cause of a decision [32] or the degree to which a human can consistently predict the model's result [33]. The IDC is interpretable in that the learned value and policy functions are an approximation to the optimal value and the optimal action of the constrained OCPs. The path selection result is expected to be the candidate path with the least optimal objective value among all the candidates. And the path tracking result is expected to be the optimal action of the OCP corresponding to the optimal path. These results can be understood from the construction of the OCPs and exactly predicted using MPC methods.

III. Static Path Planning

This module aims to generate multiple candidate paths for optimal tracking of the lower layer while maintaining high computational efficiency. For that purpose, given the road map, we generate multiple paths only considering static traffic information, such as road topology, traffic rules, etc. Each path is a set of points that composed of vertical coordinate p_x^{ref}, horizontal coordinate p_y^{ref}, and its direction ϕ^{ref}. We summarize two strategies to generate candidate paths. One is the pregenerating method, in which these paths are produced in advance and do not change with the riding of the ego vehicle. A convenient way to generate these paths is to discretize the centerline of each lane, where $(p_x^{\text{ref}}, p_y^{\text{ref}})$ is the coordinate of each discrete lane point and the direction ϕ^{ref} is aligned with the lane direction. In scenes without lanes, such as intersections, a virtual lane is constructed to connect each allowable pair of entrance lanes and exit lanes. The other strategy is the real-time generating method, where the paths are planned in real time and

always start from the ego vehicle. For this one, the third-order Bezier curve connects the ego vehicle and the centerline of the available target lanes. The control points enforce the direction of the start point and the end point to align with the ego heading and the target lane direction, respectively. The former is convenient and straightforward to be directly embedded in the map and has higher efficiency because it only needs to read the prestored paths every step. In contrast, the latter has higher flexibility and may conduct more complex driving behaviors, but its online computing efficiency will be lower than the former. Both methods will be much more efficient than currently existing methods as they generate regular candidate paths without considering collision avoidance with dynamic obstacles. The resulting paths do not incorporate dynamic information, such as global routing no matter which strategy we use. Therefore, an upstream routing module is necessary to further provide local path filtering for the current ego vehicle state before being sent to the lower layer. For example, we would only consider the left-turn paths in the lower layer when the left-turn instruction is given; or we would only consider the centerlines of the neighbor lanes when the lane-change request is launched. Note that these filtered paths only serve as references for the lower layer. The actual traveled path can be essentially different due to further considerations on dynamic obstacles.

As for the expected velocity, we heuristically assign different speed levels concerning road regions, traffic signals, and traffic rules, such as speed limits or stop signs, which can be quickly designed according to human knowledge. An example is shown in Fig. 1. The expected velocity provides a goal for the lower layer to track like the candidate paths. Still, it does not necessarily follow strictly so that the ego vehicle seeks to minimize the tracking error while satisfying safety constraints. It can be simply a fixed value, and the lower layer will still always learn a driving policy to balance the safety requirements and tracking errors.

IV. Dynamic Optimal Tracking

A. Problem Formulation

$$\begin{aligned}\min_{u_{i|t},i=0:T-1}\quad & J=\sum_{i=0}^{T-1}\left(x_{i|t}^{\text{ref}}-x_{i|t}\right)^{\top}Q\left(x_{i|t}^{\text{ref}}-x_{i|t}\right)+u_{i|t}^{\top}Ru_{i|t}\\ \text{s.t.}\quad & x_{i+1|t}=F_{\text{ego}}\left(x_{i|t},u_{i|t}\right)\\ & x_{i+1|t}^{j}=F_{\text{pred}}\left(x_{i|t}^{j}\right)\\ & \left(x_{i|t}-x_{i|t}^{j}\right)^{\top}M\left(x_{i|t}-x_{i|t}^{j}\right)\geq D_{\text{veh}}^{\text{safe}}\\ & \left(x_{i|t}-x_{i|t}^{\text{road}}\right)^{\top}M\left(x_{i|t}-x_{i|t}^{\text{road}}\right)\geq D_{\text{road}}^{\text{safe}}\\ & x_{i|t}\leq L_{\text{stop}},\ \text{if light=red}\\ & x_{0|t}=x_t,\ x_{0|t}^{j}=x_t^{j},\ u_{0|t}=u_t\\ & i=0:T-1,j\in I.\end{aligned}\tag{1}$$

In each time step t, provided multiple candidate paths generated, the lower layer is designed to first select an optimal path $\tau^*\in\Pi$ according to a particular criterion, where Π denotes a collection of N candidate paths. And then, it obtains the control quantities u_t by optimizing a finite-horizon-constrained OCP, in which the objective is to minimize the tracking error as well as the control energy, and the constraints are to keep a safe distance from obstacles, as shown in (1). T is the prediction horizon, $x_{i|t}$ and $u_{i|t}$ are the ego vehicle state and control in the virtual predictive time step i starting from the current time step t, where "virtual" means the time steps within the predictive horizon. $x_{i|t}^{\text{ref}}$ and $x_{i|t}^{\text{road}}$ are the closest point from $x_{i|t}$ on the selected reference τ^* and on the road edge, respectively. $x_{i|t}^{j}$ is the state of the jth vehicle in the interested vehicle set I. Q, R, and M are positive-definite weighting matrices. F_{ego} represents the bicycle vehicle dynamics with linear tire model [34]. F_{pred}, on the other hand, is the surrounding vehicle prediction model, which is a simple deduction from the current state with constant speed and turning curvature depending on the driving scenarios. Besides, $D_{\text{veh}}^{\text{safe}}$ and $D_{\text{road}}^{\text{safe}}$ denote the safe distance from other vehicles and the road edge. L_{stop} is the position of the stop line. Note that in (1), the virtual states are all produced by the dynamics model and the prediction model except that the $x_{0|t}$ is assigned with the current real state x_t. The variables and functions in (1) are further defined as follows:

$$\begin{aligned}x_{i|t}^{\text{ref}}&=\left[p_{\text{x}}^{\text{ref}},p_{\text{y}}^{\text{ref}},v_{\text{lon}}^{\text{ref}},0,\phi^{\text{ref}},0\right]_{i|t}^{\top}\\ x_{i|t}^{\text{road}}&=\left[p_{\text{x}}^{\text{road}},p_{\text{y}}^{\text{road}},0,0,0,0\right]_{i|t}^{\top}\\ x_{i|t}&=\left[p_{\text{x}},p_{\text{y}},v_{\text{lon}},v_{\text{lat}},\phi,\omega\right]_{i|t}^{\top}\\ x_{i|t}^{j}&=\left[p_{\text{x}}^{j},p_{\text{y}}^{j},v_{\text{lon}}^{j},0,\phi^{j},0\right]_{i|t}^{\top}\\ u_{i|t}&=\left[\delta,a\right]_{i|t}^{\top}\end{aligned}\tag{2}$$

where p_{x} and p_{y} are the position coordinates, for ego and other vehicles, it is the position of their respective center of gravity (CG), v_{lon} and v_{lat} are the longitudinal and lateral velocities, ϕ is the heading angle, ω is the yaw rate, and δ and a are the front-wheel angle and the acceleration commands, respectively.

The optimal path is chosen as the one with the best tracking performance while satisfying the safety requirements, i.e.,

$$\tau^*=\arg\min_{\tau}\left\{J_{\tau}^*|\tau\in\Pi\right\}\tag{3}$$

where J_{τ}^* is the optimal cost of the path τ. This means that for each path candidate $\tau\in\Pi$, we ought to construct such an OCP (1) and to optimize it to obtain its optimal cost J_{τ}^*. Such a criterion of path selection is consistent with the objective of the OCP (1) in the sense that it optimizes the problem concerning paths within the candidate path set, compared to control quantities in the original optimization.

In such a framework of selecting and tracking, the lower layer can well determine a control quantity that yields good driving efficiency and, meanwhile, the safety guarantee. The OCPs can be solved by classical control methods, such as Pontryagin's maximum principle or the direct shooting method. However, each OCP has a high-dimensional nonlinear system with hundreds of variables and thousands of constraints, leading to high time complexity for online optimization. Therefore, we employ RL to unload the online

optimization burden. Specifically, we show that this framework naturally corresponds to the actor–critic architecture of RL, where the critic, with the function of judging state goodness, can be served as the path selector while the actor is in charge of action output and used for tracking. With a paradigm of offline training and online application, we can almost nearly eliminate the computation burden in the lower layer. A similar idea for online computing efficiency can be found in [35] and [36].

B. Offline Training

1) Preliminaries: RL considers an MDP where an agent interacts with its environment with the aim of learning utility-minimizing behavior. At each discrete-time step, with a given state $s \in \mathcal{S}$, the agent selects actions $u \in \mathcal{U}$ with respect to its policy $\pi : \mathcal{S} \to \mathcal{U}$, receiving an utility $l(s, u)$ and the new state of the environment $s' = f(s, u)$, where $l : \mathcal{S} \times \mathcal{U} \to \mathbb{R}$ denotes the utility function and $f : \mathcal{S} \times \mathcal{U} \to \mathcal{S}$ denotes the system model. The value function $v^\pi : \mathcal{S} \to \mathbb{R}$ is defined as the expected sum of utilities obtained from the input state s and using π, i.e., $v^\pi(s) = \{\sum_{t=0}^{T-1} l_t | s_0 = s\}$. RL aims to find the optimal parameterized policy π_θ called actor to minimize the expected sum of utilities, i.e., $\mathbb{E}_{s\sim d}\{v^\pi(s)\}$, where d is a state distribution. Also, it optimizes a parameterized value function V_w called critic to approximate the true value function. θ and w are parameters to be optimized.

2) Complete RL Problem Formulation: We aim to solve the path selecting and path tracking problems (1) and (3) by RL. Notice that there are several significant differences between the OCP (1) and RL problems, originating from the online or offline optimizations. The OCP, designed for online optimization, seeks to find a single optimal control quantity of a single state given a specific path at time step t. On the other hand, RL problems aim to solve the parameters of the optimal actor and critic, i.e., θ and w, in an offline style. In addition, the objective function and constraints of RL are not about a single state but distribution in the state space. The state s should contain the necessary information to determine driving actions. Except for the knowledge of the ego vehicle and surrounding vehicles, we also incorporate the path information as a part of states to obtain a policy that can handle tracking tasks for different paths, denoted by $s_t \leftarrow \{\tau, x_t, x_t^j, j \in I\}$. The resulting RL problem is shown as follows:

$$\begin{aligned}
\min_{\theta}\ & J_{\text{actor}} = \mathbb{E}_{s_{0|t}}\left\{\sum_{i=0}^{T-1} l\big(s_{i|t}, \pi_\theta\big(s_{i|t}\big)\big)\right\} \\
\text{s.t.}\ & s_{i+1|t} = f\big(s_{i|t}, \pi_\theta\big(s_{i|t}\big)\big) \\
& g_e\big(s_{i|t}\big) \geq 0, e \in E \\
& s_{0|t} = s_t \leftarrow \left\{\tau, x_t, x_t^j, j \in J\right\} \sim d, \\
& i = 0 : T-1 \\
\min_{w}\ & J_{\text{critic}} = \mathbb{E}_{s_{0|t}}\left\{\left(\sum_{i=0}^{T-1} l\big(s_{i|t}, \pi_\theta\big(s_{i|t}\big)\big) - V_w\big(s_{0|t}\big)\right)^2\right\} \\
\text{s.t.}\ & s_{i+1|t} = f\big(s_{i|t}, \pi_\theta\big(s_{i|t}\big)\big) \\
& s_{0|t} = s_t \sim d, \\
& i = 0 : T-1
\end{aligned} \tag{4}$$

$$\begin{aligned}
\min_{\theta}\ & J_{\text{actor}} = \mathbb{E}_{s_{0|t}}\left\{\sum_{i=0}^{T-1} l\big(s_{i|t}, \pi_\theta\big(s_{i|t}\big)\big)\right\} \\
\text{s.t.}\ & s_{i+1|t} = f\big(s_{i|t}, \pi_\theta\big(s_{i|t}\big)\big) \\
& g_e\big(s_{i|t}\big) \geq 0, e \in E \\
& s_{0|t} = s_t \leftarrow \left\{\tau, x_t, x_t^j, j \in J\right\} \sim d, \\
& i = 0 : T-1
\end{aligned} \tag{5}$$

where $l(s_{i|t}, \pi_\theta(s_{i|t})) = (x_{i|t}^{\text{ref}} - x_{i|t})^\top Q(x_{i|t}^{\text{ref}} - x_{i|t}) + \pi_\theta^\top(s_{i|t}) R \pi_\theta(s_{i|t})$. f is an aggregation of the F_{ego} and F_{pred}. $g_e(s_{i|t}), e \in E$ denotes all the constraints about the state $s_{i|t}$, including that with other vehicles, road, and traffic rules. π_θ and V_w are generally in form of NNs. Detailed design of MDP elements is in Section V-B, including state, action, utility, constraints, etc. From the results of [37], given overparameterized NNs, the optimal policy π_{θ_*} of (4) maps to an optimal action of the original OCP (1) with arbitrary initial state s_t. Consequently, the optimal value J^* from (1), of course, would be equal to the one mapped by the optimal value function V_{w_*} of (5) from s_t, i.e.,

$$\begin{aligned}
u_t^* &= \pi_{\theta_*}(s_t) \quad \forall s_t \in \mathcal{S} \\
J^* &= V_{w_*}(s_t) \quad \forall s_t \in \mathcal{S}.
\end{aligned} \tag{6}$$

In other words, the optimal policy and value function can output the optimal control and value under arbitrary states, i.e., arbitrary combinations of paths, ego state, and surrounding vehicle states.

3) Generalized Exterior Point Method: We adopt the policy iteration framework to solve the converted RL problem, wherein policy evaluation and policy improvement are alternatively performed to update the critic and the actor. Since the critic update is an unconstrained problem that can be optimized by ordinary gradient descent methods, we mainly focus on the actor update, which is quite challenging because of its large-scale parameter space, nonlinear property, and the infinite number of state constraints. To tackle this, we propose a model-based RL algorithm called GEP adapted from the one in the optimization field. It first transforms the constrained problem (4) into an unconstrained one by the exterior penalty function, shown as follows:

$$\begin{aligned}
\min_{\theta}\ & J_p = J_{\text{actor}} + \rho J_{\text{penalty}} \\
& = \mathbb{E}_{s_{0|t}}\left\{\sum_{i=0}^{T-1} l\big(s_{i|t}, \pi_\theta\big(s_{i|t}\big)\big)\right\} + \rho \mathbb{E}_{s_{0|t}}\left\{\sum_{i=0}^{T-1} \varphi_i(\theta)\right\} \\
\text{s.t.}\ & s_{i+1|t} = f\big(s_{i|t}, \pi_\theta\big(s_{i|t}\big)\big) \\
& \varphi_i(\theta) = \sum_{e\in E}\big[\max\big\{0, -g_e\big(s_{i|t}\big)\big\}\big]^2 \\
& s_{0|t} = s_t \sim d, \\
& i = 0 : T-1
\end{aligned} \tag{7}$$

where φ is the penalty function and ρ is the penalty factor. After that, we alternatively optimize the policy parameters by performing m iterations of gradient descent and increase the penalty factor by multiplying a scalar $c > 1$. We call the former the optimizing procedure and the latter the amplifying procedure. Different from the exterior point method, the optimizing procedure of GEP does not necessarily find the optimal solution of the unconstrained problem (7), but we will prove that GEP still converges to the optimal policy under certain conditions. GEP is simple to implement and is powerful in dealing with large-scale parameter space facilitated by

Algorithm 1 Dynamic Optimal Tracking-Offline Training

Initialize: critic network V_w and actor network π_θ with random paramaters w, θ, buffer $\mathcal{B} \leftarrow \emptyset$, learning rates β_w, β_θ, penalty factor $\rho = 1$, penalty amplifier c, update interval m
for each iteration i **do**
  // Sampling (from environment)
  Randomly select a path $\tau \in \Pi$, initialize ego state x_t and vehicle states $x_t^j, j \in I$
  for each environment step **do**
    $s_t \leftarrow \{\tau, x_t, x_t^j, j \in I\}$
    $\mathcal{B} \cup \{s_t\}$
    $u_t = \pi_\theta(s_t)$
    Apply u_t to observe x_{t+1} and $x_{t+1}^j, j \in I$
  end for

  // Optimizing (GEP)
  Fetch a batch of states from $\mathcal{B}$, compute J_{critic} and J_p by f and π_θ
  PEV: $w \leftarrow w - \beta_w \nabla_w J_{\text{critic}}$
  PIM: if $i \mod m$:$\rho \leftarrow c\rho$; $\theta \leftarrow \theta - \beta_\theta \nabla_\theta J_p$
end for

the gradient descent technique. Besides, from the form of (7), numerous state constraints can be handled naturally by regarding the constraint violation as a term of utility multiplied by ρ. The training pipeline is shown in Algorithm 1.

Next, we present the convergence proof of GEP. We say a "round" completes when an optimizing procedure is finished. Since the optimizing procedure only improves (7) a fixed number of times to obtain a fair solution but does not necessarily find the optimal one, we first give an assumption about how well the solution is.

Assumption 1: After the round k completes, we have the penalty factor ρ_k and an optimized policy parameter θ_k. We assume that θ_k satisfies

$$J_p(\theta_k, \rho_k) \le \min_\theta J_p(\theta, \rho_k) + \Delta_k, k = 1, 2, \ldots \tag{8}$$

where $\Delta_k \ge 0, k \ge 1$ is a positive nonincreasing sequence that has finite series, i.e., $\Delta_k \ge \Delta_{k+1}$, $\sum_{i=0}^{\infty} \Delta_i < \infty$.

Assumption 1 describes that with the convergence of NNs, the gap between the solution of the optimizing procedure and the optimal one is gradually eliminated, i.e., $\lim_{k\to\infty} \Delta_k = 0$, as indicated by the finite series. About the gap, we have the following Lemma.

Lemma 1: There exists a positive nonincreasing sequence $\delta_k, k \ge 1$ that satisfies

$$\begin{aligned} &\Delta_k = \delta_k - \delta_{k+1} \\ &\delta_k \ge \delta_{k+1}, \lim_{k\to\infty} \delta_k = 0. \end{aligned} \tag{9}$$

Proof: We can construct such a sequence by setting

$$\delta_1 = \sum_{i=1}^{\infty} \Delta_i, \quad \delta_{k+1} = \delta_k - \Delta_k, k \ge 1. \tag{10}$$

Then, $\delta_1 < 0$ holds by Assumption 1 and the convergence of δ_k naturally holds by

$$\lim_{k\to\infty} \delta_k = \lim_{k\to\infty} \delta_1 - \sum_{i=1}^{k-1} \Delta_i = \delta_1 - \lim_{k\to\infty} \sum_{i=1}^{k-1} \Delta_i = 0. \tag{11}$$

■

Next, we first prove the following two lemmas about the unconstrained objective.

Lemma 2: For the solution sequence generated after each round $\{\theta_k\}$, we have

$$J_p(\theta_{k+1}, \rho_{k+1}) - \delta_{k+1} \ge J_p(\theta_k, \rho_k) - \delta_k. \tag{12}$$

Proof: By $J_p(\theta, \rho) = J_{\text{actor}}(\theta) + \rho J_{\text{penalty}}(\theta)$ and $\rho_{k+1} > \rho_k$

$$\begin{aligned} J_p(\theta_{k+1}, \rho_{k+1}) &= J_{\text{actor}}(\theta_{k+1}) + \rho_{k+1} J_{\text{penalty}}(\theta_{k+1}) \\ &\ge J_{\text{actor}}(\theta_{k+1}) + \rho_k J_{\text{penalty}}(\theta_{k+1}) \\ &= J_p(\theta_{k+1}, \rho_k). \end{aligned} \tag{13}$$

Then, by Assumption 1, for $\forall\theta$, $J_p(\theta, \rho_k) \ge \min_\theta J_p(\theta, \rho_k) \ge J_p(\theta_k, \rho_k) - \Delta_k$, thus we have $J_p(\theta_{k+1}, \rho_k) \ge J_p(\theta_k, \rho_k) - \Delta_k$, therefore

$$\begin{aligned} &J_p(\theta_{k+1}, \rho_{k+1}) \ge J_p(\theta_k, \rho_k) - \Delta_k \\ &J_p(\theta_{k+1}, \rho_{k+1}) - \delta_{k+1} \ge J_p(\theta_k, \rho_k) - \delta_k. \end{aligned} \tag{14}$$

■

Lemma 3: Suppose $\theta_* = \arg\min_\theta J_{\text{actor}}(\theta)$, then for $\forall k \ge 1$

$$J_{\text{actor}}(\theta_*) - \delta_{k+1} \ge J_p(\theta_k, \rho_k) - \delta_k \ge J_{\text{actor}}(\theta_k) - \delta_k. \tag{15}$$

Proof: Because θ_* is the optimal solution of the problem (4), it has $J_{\text{penalty}}(\theta_*) = 0$. Then, from Assumption 1, the first inequality is obtained as follows:

$$\begin{aligned} &J_{\text{actor}}(\theta_*) = J_p(\theta_*, \rho_k) \ge \min_\theta J_p(\theta, \rho_k) \ge J_p(\theta_k, \rho_k) - \Delta_k \\ &J_{\text{actor}}(\theta_*) - \delta_{k+1} \ge J_p(\theta_k, \rho_k) - \delta_k. \end{aligned} \tag{16}$$

The second inequality is obtained by $\rho_k J_{\text{penalty}}(\theta_k) \ge 0$

$$J_p(\theta_k, \rho_k) = J_{\text{actor}}(\theta_k) + \rho_k J_{\text{penalty}}(\theta_k) \ge J_{\text{actor}}(\theta_k). \tag{17}$$

■

The convergence can be revealed by Theorem 1.

Theorem 1: Assume that J_{actor} and J_{penalty} are continuous functions defined on the parameter space. Suppose $\{\theta_k\}$ is the solution sequence generated after each round. The limit of any of its convergent subsequence is the optimal solution.

Proof: Suppose $\{\theta_{k_j}\}$ is an arbitrary convergent subsequence of $\{\theta_k\}$ with the limit $\bar{\theta}$. By the continuity of J_{actor}, $\lim_{k_j\to\infty} J_{\text{actor}}(\theta_{k_j}) = J_{\text{actor}}(\bar{\theta})$. Define $J_{\text{actor}}^* = \min_\theta J_{\text{actor}}(\theta)$ as the optimal value of the problem (4). From Lemmas 2 and 3, we can see that $\{J_p(\theta_{k_j}, \rho_{k_j}) - \delta_{k_j}\}$ is a nonincreasing sequence with upper limit J_{actor}^*, therefore

$$\lim_{k_j\to\infty} \left(J_p\left(\theta_{k_j}, \rho_{k_j}\right) - \delta_{k_j}\right) = \lim_{k_j\to\infty} J_p\left(\theta_{k_j}, \rho_{k_j}\right) = J_p^* \le J_{\text{actor}}^*. \tag{18}$$

Then, because $J_p(\theta_{k_j}, \rho_{k_j}) = J_{\text{actor}}(\theta_{k_j}) + \rho_{k_j} J_{\text{penalty}}(\theta_{k_j})$

$$\lim_{k_j\to\infty} \rho_{k_j} J_{\text{penalty}}\left(\theta_{k_j}\right) = J_p^* - J_{\text{actor}}\left(\bar{\theta}\right) \tag{19}$$

by $J_{\text{penalty}}(\theta_{k_j}) \geq 0$, $\rho_{k_j} \to \infty$ and the continuity of J_{penalty}, we have

$$\lim_{k_j \to \infty} J_{\text{penalty}}(\theta_{k_j}) = J_{\text{penalty}}(\bar{\theta}) = 0 \tag{20}$$

which indicates that the limit $\bar{\theta}$ is a feasible solution. Furthermore, by Lemma 3, $J_{\text{actor}}(\theta_{k_j}) \leq J^*_{\text{actor}}$, together with the continuity of J_{actor}, we have

$$J_{\text{actor}}(\bar{\theta}) \leq J^*_{\text{actor}} \Rightarrow J_{\text{actor}}(\bar{\theta}) = J^*_{\text{actor}} \tag{21}$$

where the equation holds by the definition of J^*_{actor}. Equation (21) indicates the optimality of the limit $\bar{\theta}$. ■

C. Online Application

Ideally, we expect an optimal policy to output the optimal action within the safety action space at any given point in the state space. Unfortunately, it is impossible to acquire such a policy in theory and practice. First, the converted RL problem (4) enforces constraints on every single state point, leading to an infinite number of constraints in the continuous state space. Nevertheless, when we solve the equivalent unconstrained problem (7), we approximate the expectation by an average of samples, which means we only consider finite constraints of the set of samples that can vary in different iterations. That is why there is no strict safety guarantee of the policy but only an approximately safe performance. Second, the condition of (6) that the approximation function has infinite fitting power cannot be established in practice, resulting in a suboptimal solution without a safety guarantee. To ensure safety performance, we adopt a multistep safety shield after the output of the policy.

1) Multistep Safety Shield: The safety shield aims to find the nearest actions in the safe action space in state s_t, which is formulated as a quadratic programming problem

$$u_t^{\text{safe}} = \begin{cases} u_t^*, & \text{if} \quad u_t^* \in \mathcal{U}_{\text{safe}}(s_t) \\ \arg\min_{u \in \mathcal{U}_{\text{safe}}(s_t)} \|u - u_t^*\|_2^2, & \text{else} \end{cases} \tag{22}$$

where u_t^* is the policy output. Rather than designing $\mathcal{U}_{\text{safe}}(s_t)$ to guarantee the safety of only the next state, we design it to guarantee that the next n_{ss} prediction states are safe, i.e., collision free with the surrounding vehicles and road edges. Formally

$$\mathcal{U}_{\text{safe}}(s_t) = \{u_t | g_e(s_{i|t}) \geq 0, i = 1, \ldots, n_{ss}, e \in E\}. \tag{23}$$

2) Algorithm for Online Application: We first construct a set of states for different paths given the trained policy and value functions. Then, we pass them to the trained value function to select the one with the lowest value, which is next passed to the trained policy to obtain the optimal control, as summarized in Algorithm 2.

Algorithm 2 Dynamic Optimal Tracking-Online Application

Initialize: Path set Π from upper layer, trained critic network V_{w^*} and actor network π_{θ^*}, λ, ego state x_t and vehicle states $x_t^j, j \in I$
for each environment step **do**
  // Selecting
  for each $\tau \in \Pi$ **do**
    $s_{t,\tau} \leftarrow \{\tau, x_t, x_t^j, j \in I\}$
    $V_\tau^* = V_{w^*}(s_{t,\tau})$
  end for
  $\tau^* = \arg\min_\tau \{V_\tau^* | \tau \in \Pi\}$

  // Tracking
  $s_t \leftarrow \{\tau^*, x_t, x_t^j, j \in I\}$
  $u_t^* = \pi_{\theta^*}(s_t)$
  Calculate u_{safe}^* by (22)
  Apply u_{safe}^* to observe x_{t+1} and $x_{t+1}^j, j \in I$
end for

V. Simulation Verification

A. Scenarios

We carried out simulation experiments on two scenarios. The first scenario is a four-lane highway shown in Fig. 2(a). We mainly investigate the basic functionalities of the learned policy in this scenario. The second one is a regular signalized four-way intersection, where the roads in different directions are all the six-lane dual carriageway, as shown in Fig. 2. Each intersection entrance has three lanes for turning left, going straight, and turning right, respectively.

Moreover, a two-phase traffic signal controls the traffic flow of turning left and going straight. In this scenario, we verify our algorithm in three tasks: 1) turn left; 2) go straight; and 3) turn right. In each task, the ego vehicle is initialized randomly from the south bound and is expected to drive safely and efficiently to pass the intersection. For both scenarios, we generate a dense traffic flow of 800 vehicles per hour on each lane with SUMO [38]. These vehicles are controlled by the car-follow and lane-change models in the SUMO, producing a variety of traffic behaviors.

B. Implementation of Training Algorithm

1) State Design: As mentioned in Section IV-B2, the state should be designed to include information of the ego vehicle, the surrounding vehicles, and the reference path, i.e.,

$$s = \left[s^{\text{ego}}, s^{\text{other}}, s^{\text{ref}}\right] \tag{24}$$

where s^{ego} is the ego vehicle state x defined in Section IV-A, and s^{other} is the concatenation of the interested surrounding vehicle list I

$$s^{\text{other}} = \left[p_x^j, p_y^j, \phi^j, v_{\text{lon}}^j\right]_{j \in I}. \tag{25}$$

I is an ordered list of vehicles that have potential conflicts with the ego, which is different in different scenarios and tasks. For highway scenario

$$I = [\text{LF}, \text{LB}, \text{CF}, \text{CB}, \text{RF}, \text{RB}] \tag{26}$$

where L, C, and R represent the left/current/right lane and F and B means the front/back vehicle. For the intersection scenario, take the left-turn task as an example

$$I = [\text{SW1}, \text{SW2}, \text{SN1}, \text{SN2}, \text{NS1}, \text{NS2}, \text{NW1}, \text{NW2}] \tag{27}$$

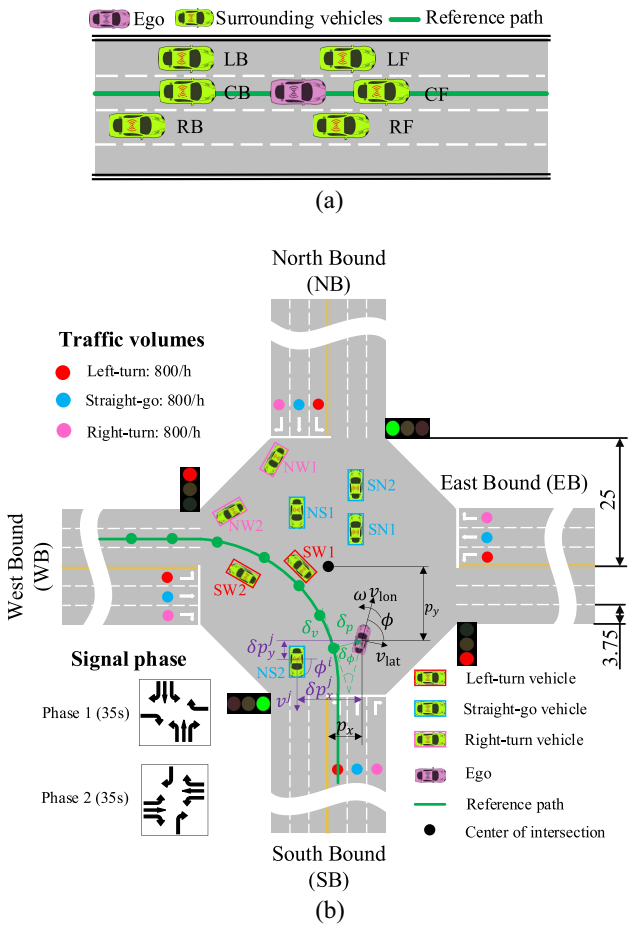


Fig. 2. Experimental scenarios and the state design. (a) Highway. (b) Intersection.

where each slot is named by their respect route start and end [namely, South (S), North (N), West (W), East (E)], as well as the order on that (1 or 2), e.g., "SW1" means the first vehicle drives from south to west, as shown in Fig. 2. Correspondingly, one can define the straight-go and right-turn tasks similarly. The information of the reference s_t^{ref}, however, is designed implicitly by the tracking errors concerning the position, the heading angle, and the velocity

$$s^{\text{ref}} = \left[\delta_p, \delta_\phi, \delta_v\right] \tag{28}$$

where δ_p is the position error, $|\delta_p| = \sqrt{(p_x - p_x^{\text{ref}})^2 + (p_y - p_y^{\text{ref}})^2}$, and $\text{sign}(\delta_p)$ is positive if the ego is on the left side of the reference path, or else is negative. $\delta_\phi = \phi - \phi^{\text{ref}}$ is the error of heading angle, and $\delta_v = v_{\text{lon}} - v_{\text{lon}}^{\text{ref}}$ is the velocity error. The overall state design is illustrated in Fig. 2. The input dimension of the policy network in different scenarios and tasks is summarized in Table I.

2) Action Design: The action u is the control quantities of the ego vehicle consisting of the expected front-wheel angle δ and acceleration a, i.e.,

$$u = [\delta, a]^\top. \tag{29}$$

Note that the action output by the policy is normalized in the range $[-1, 1]$ by the tanh activation. The front-wheel angle is further scaled to $[-0.4, 0.4]$ rad while the acceleration is further scaled and shifted to $[-3, 1.5]$ m/s^2.

TABLE I
INPUT DIMENSION OF THE POLICY NETWORK IN DIFFERENT TASKS

State	Highway	Intersection		
		Left-turn	Straight-go	Right-turn
s^{ego}	6	6	6	6
s^{other}	4×6=24	4×8=32	4×9=36	4×5=20
s^{ref}	3	3	3	3
s (total)	33	41	45	29

3) Utility and Objective Design: The utility is already defined in Section IV-B2 for equivalent problem conversion, as shown as follows:

$$l(s, u) = \left(x^{\text{ref}} - x\right)^\top Q\left(x^{\text{ref}} - x\right) + u^\top Ru. \tag{30}$$

Here, we further design the weighting matrices in it as $Q = \text{diag}(0.04, 0.04, 0.01, 0.01, 0.1, 0.02)$ and $R = \text{diag}(0.1, 0.005)$. Expanding the utility we have the following result:

$$l(s, u) = 0.04\delta_p^2 + 0.01\delta_v^2 + 0.1\delta_\phi^2 + 0.1\delta^2 + 0.005a^2. \tag{31}$$

The utility indicates the tracking errors and the control energy consumption yielded in a time step. The tracking errors include the weighted squared errors in position, heading, and speed. The control energy, on the other hand, contains the weighted squared quantities in terms of steering and acceleration.

As indicated by (4), the objective is to minimize the sum of utilities within the predictive horizon, i.e., the sum of tracking errors and control energy consumption. The predictive horizon T is set to be 25, which is 2.5 s in practice.

4) Constraint Design: Slightly different from the one in (1), we further refine the constraint in a way that represents the ego vehicle and each of the surrounding vehicles by two covered circles. Then, in each time step, we impose four constraints for each vehicle between each center of the ego circle and that of the other vehicle. Similar are the constraints between the ego and the road edge.

5) Training Settings: We implement the offline training algorithm (Algorithm 1) in an asynchronous learning architecture proposed in [39]. For value function and policy, we use a multilayer perceptron (MLP) with two hidden layers, consisting of 256 units per layer, with exponential linear units (ELUs) in each layer [40]. The Adam method [41] with a polynomial decay learning rate is used to update all the parameters. Specific hyperparameter settings are listed in Table II. We train five different runs with different random seeds on a single computer with a 2.4 GHz 50 core Inter Xeon CPU, with evaluations every 100 iterations.

C. Simulation Results

The static path planning result is shown in Fig. 3(a). We also demonstrate the tracking and safety performances at the intersection during the training process in Fig. 3, indicated

TABLE II
DETAILED HYPERPARAMETERS

Hyperparameters	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	5e5
Batch size	1024
Policy learning rate	Linear decay 3e-4 $\rightarrow$ 1e-5
Value learning rate	Linear decay 8e-4 $\rightarrow$ 1e-5
Penalty amplifier c	1.1
Total iteration	200000
Update interval m	10000
Safety shield n_{ss}	5
Number of Actors	4
Number of Buffers	4
Number of Learners	30

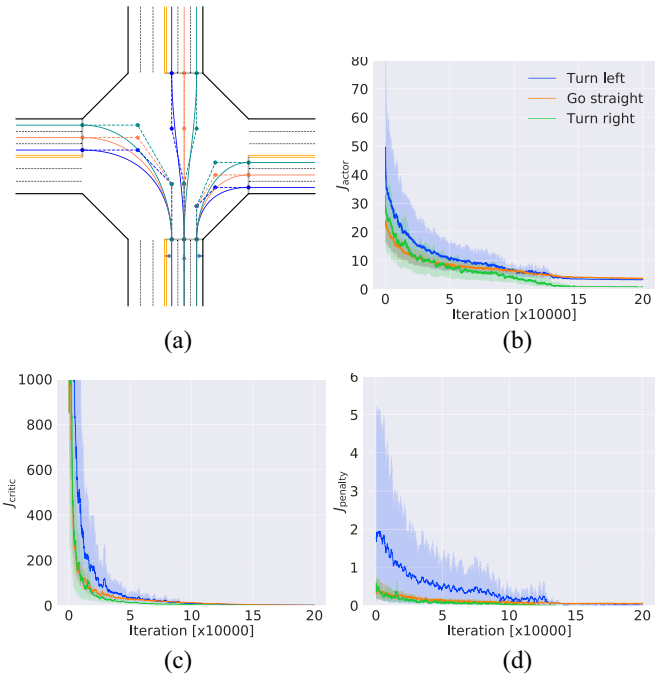


Fig. 3. Results of static path planning and dynamic optimal tracking. (a) Planned paths for each task. (b) Tracking performance during the training process. (c) Safety performance during the training process. (d) Value loss during the training process. For (b)–(d), the solid lines correspond to the mean and the shaded regions correspond to 95% confidence interval over five runs.

by J_{actor} and J_{penalty}, respectively, and the value loss J_{critic} to exhibit the performance of the value function.

Along the training process, the policy loss, the penalty, and the value loss decrease consistently for all the tasks, indicating an improving tracking and safety performance. Especially, the penalty and value loss decrease to zero approximately, proving the effectiveness of the proposed RL-based solver for constrained OCPs. In addition, the convergence speed, variance across different seeds, and the final performance vary with tasks. That is, because the surrounding vehicles that have potential conflicts with the ego are different across tasks, leading to differences in task difficulty.

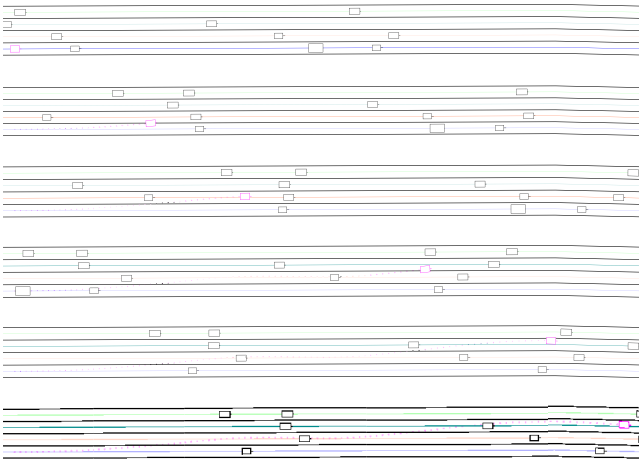


Fig. 4. Visualization of one typical episode in highway.

We visualize the decision and control process of the trained policy in highway and intersection, respectively, as shown in Figs. 4 and 5. Benefiting from the interpretability, we can explain the resulting driving behaviors. For the highway scenario, the ego is initialized on the rightmost lane and aims to drive as fast as possible under the premise of security and compliance. When the front car of the ego rides at a lower speed which may cause the collision, the ego vehicle will steer itself toward its left lane and then choose it as the optimal path and finish the first lane change. After that, the ego vehicle will accelerate along the current lane with a distance until its speed is larger than the front vehicle again. Then, the ego will choose its left lane with a sparse traffic flow and accomplishment the second lane change. Due to the high speed of the ego, a littler overshoot occurs in this process, and the ego vehicle will adjust quickly to track the current optimal path.

For the intersection scenario, we visualize one typical left-turn case where dense traffic and different phases of a traffic light are included, as shown in Fig. 5. When the traffic light is red, the ego pulls up to avoid collision to meet the safety constraints. Then, when the light turns green, the ego starts off to track the expected velocity. After entering the junction, it meets several straight-go vehicles from the opposite direction. Therefore, the ego chooses the upper path and slows down to bypass the first one. Notice that the safety shield works here to avoid potential collisions. After that, it speeds up to follow the middle path, the one with the lowest value in that case, with which it can go first and meanwhile avoid the right-turn vehicles so that it can essentially reduce the velocity tracking error. The computing time in each step is under 10 ms, making our method considerably fast to be applied in the real world.

D. Experiment 1: Performance of GEP

To verify GEP's control precision and computing efficiency on the constrained optimization problem, we compare it with the classic MPC solver. Formally, the problem (1) is defined on one particular path, and thus MPC method solves the same number of problems as that of candidate paths. Then, the optimal path is selected by the minimum cost function, and its corresponding action is used as the input signal of the ego vehicle. Here, we adopt the Ipopt solver to obtain

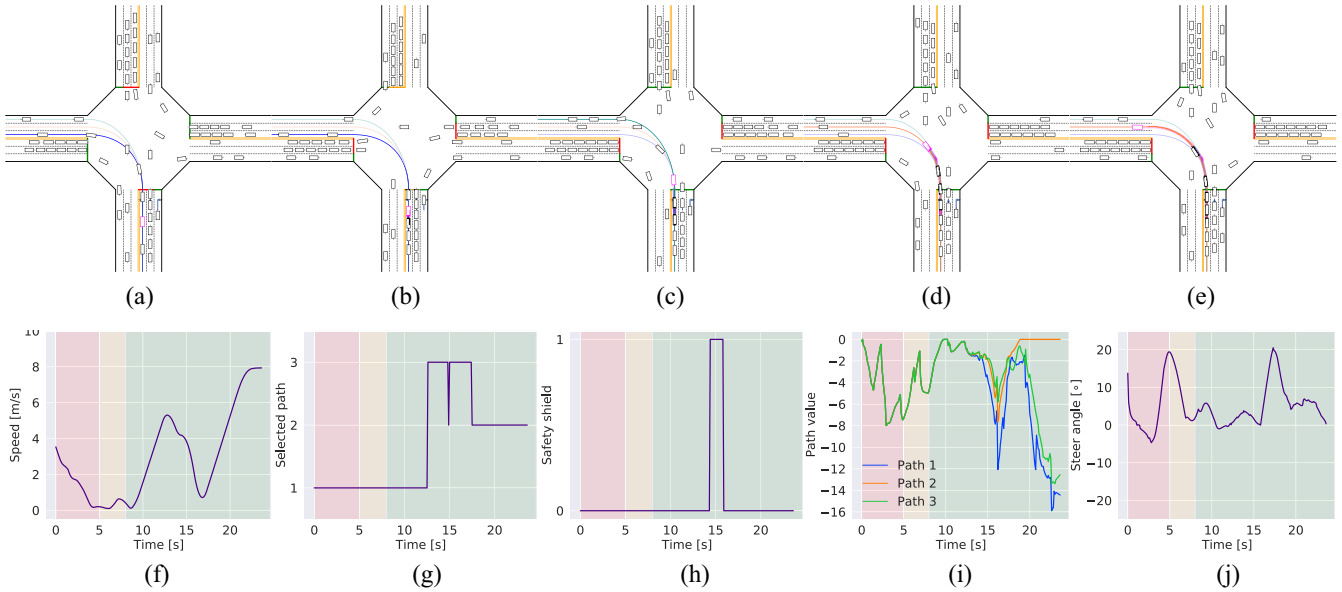


Fig. 5. Visualization of one typical episode at the intersection. (a) $t = 0.3$ s. (b) $t = 6.5$ s. (c) $t = 12.8$ s. (d) $t = 19.1$ s. (e) $t = 23.3$ s. (f) Speed. (g) Ref index. (h) Safety shield. (i) Steering angle. (j) Acceleration.

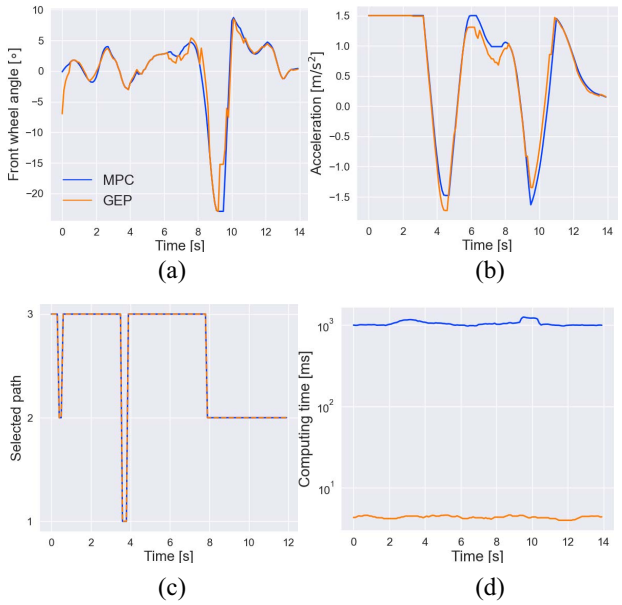


Fig. 6. Comparison with MPC. (a) Front-wheel angle. (b) Acceleration. (c) Optimal path. (d) Computing time.

the exact solution of the constrained OCP, a popular package for nonlinear optimization problems. Fig. 6 demonstrates the comparison of our algorithm on control effect and computation time. Results show that the optimal path of the two methods are identical and the output actions (steer wheel and acceleration) have similar trends, which indicates our proposed algorithm can approximate the solution of MPC with a small error. Also, it proves the solved policy and value are the optimal action and the optimal cost function of the problem (1), which is substantial evidence of the interpretability of our method. Although MPC can find the optimal solution through online optimization, its computation time also increases sharply with the number of constraints, probably violating the real-time requirements of autonomous driving.

E. Experiment 2: Comparison of Driving Performance

We compare our method with a rule-based approach and a model-free RL method. The rule-based method is a combination of graph-search-based methods and optimization-based methods [17]. It first searches a spatiotemporal trajectory using the A* algorithm. Then, a locally convex feasible driving area is designed, and optimization is applied to further smooth the trajectory with consideration of vehicle kinematics and the feasible area. It is reported to beat the other methods [42]–[44] in terms of the average planning time. The model-free RL uses a punish and reward system to learn a policy for maximizing the long-term reward (+100 if the ego vehicle passes safely; otherwise −100 whenever a collision happens) [45]. To evaluate the three algorithms, we choose six indicators, including computing time, comfort, travel efficiency, collisions, failure rate, and driving compliance. Comfort is reflected by the mean root square of lateral and longitudinal acceleration.

Travel efficiency is evaluated by the average time used to pass the intersection. Failure rate means the accumulated times of that decision signal is generated for more than one second and driving compliance shows times of breaking red light. The results of 100 simulations are shown in Table III. The rule-based method is more likely to stop and wait for other vehicles, leading to lower passing efficiency but better safety and comfort. However, it takes much more time to give a control signal in the dense traffic flow and suffers the highest failure rate. Model-free RL is eager to achieve the goal but incurs the most collisions and decision incompliance due to its lack of safety guarantee. Benefiting from the framework design, the computational efficiency of our method remains as fast as the model-free approach, and the driving performance, such as safety, compliance, and failure rate, is better than the other two approaches.

F. Experiment 3: Application of Distributed Control

We also apply our trained policies in multiple vehicles for distributed control. Our method yields a surprisingly good

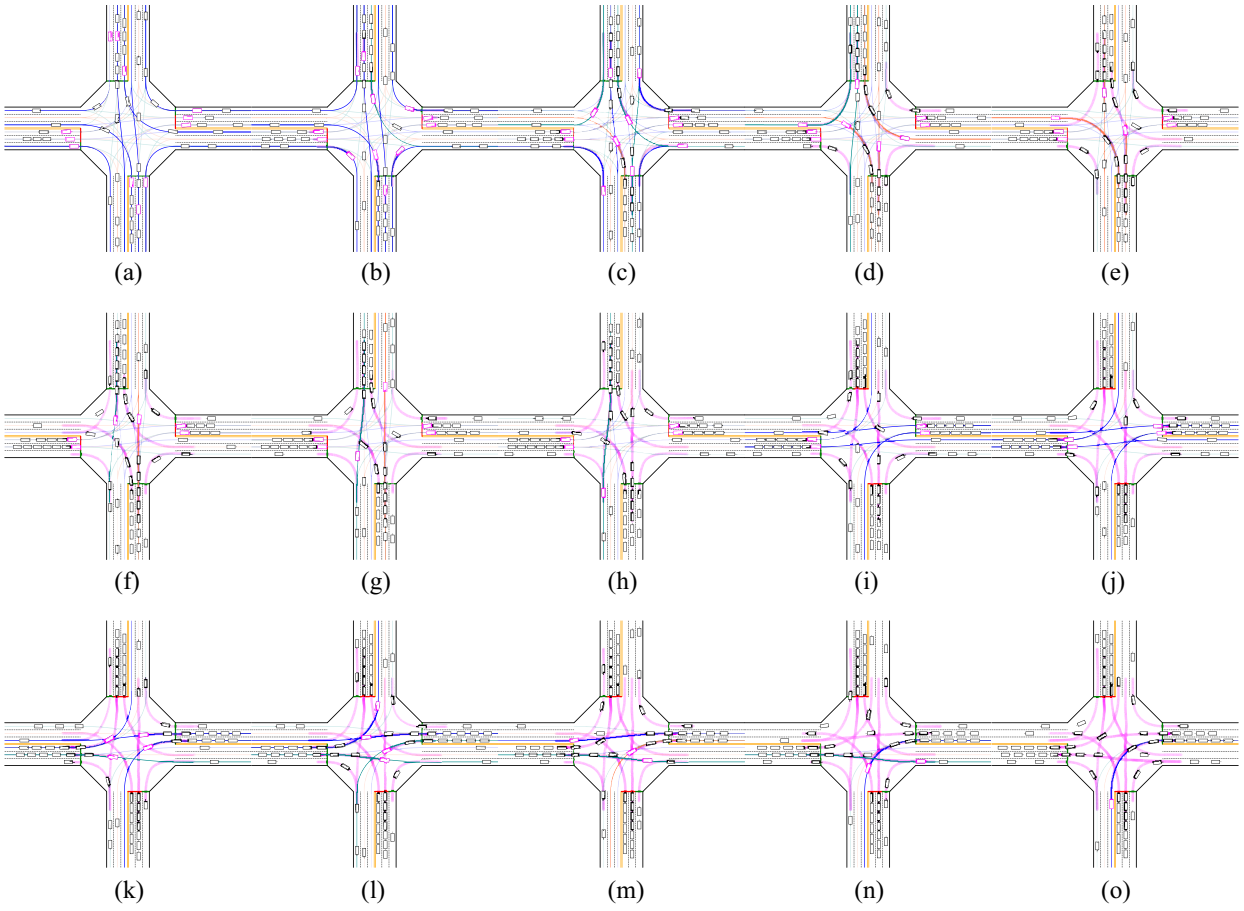


Fig. 7. Demonstration of the distributed control carried out by the trained policies. (a) $t = 0.3$ s. (b) $t = 2.8$ s. (c) $t = 5.4$ s. (d) $t = 8.0$ s. (e) $t = 10.6$ s. (f) $t = 13.2$ s. (g) $t = 15.8$ s. (h) $t = 18.4$ s. (i) $t = 28.8$ s. (j) $t = 31.4$ s. (k) $t = 34.0$ s. (l) $t = 36.6$ s. (m) $t = 39.2$ s. (n) $t = 41.8$ s. (o) $t = 44.4$ s.

TABLE III
COMPARISON OF DRIVING PERFORMANCE

	IDC (Ours)	Rule-based	Model-free RL
Computing time [ms]			
Upper-quantile	5.81	73.99	4.91
Standard deviation	0.60	36.59	0.65
Comfort index	1.84	1.41	3.21
Time to pass [s]	7.86(±3.52)	24.4(±16.48)	6.73(±3.32)
Collisions	0	0	31
Failure Rate	0	13	0
Decision Compliance	0	0	17

performance by showing a group of complex and intelligent driving trajectories (Fig. 7), which demonstrates the potential of the proposed method to be extended to the distributed control of large-scale connected vehicles. More videos are available online (https://youtu.be/J8aRgcjJukQ).

VI. TEST ON REAL-WORLD ROADS

A. Scenario and Equipment

In the real-world test, we choose an intersection of two-way streets, where the east–west street has an eight-lane dual carriageway from both directions, while the north–south street has a four-lane dual carriageway. To comply with legal requirements, we did not utilize the intersection's natural traffic flow and traffic signals. Instead, these traffic elements are designed and provided by SUMO. The control flow of different modules is shown in Fig. 8. The computer is KMDA-3211 with a 2.6 GHz Intel Core I5-6200U CPU. The ego vehicle must complete the left-turn, straight-go, and right-turn tasks under signal control and a virtual dense traffic flow.

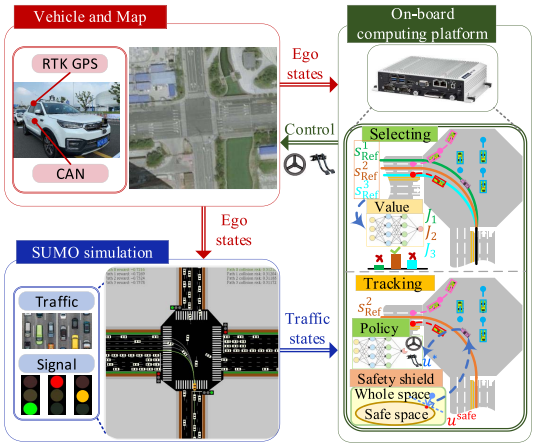


Fig. 8. Diagram of the real-world road test.

B. Experiment 1: Functionality Verification

This experiment aims to verify the functionality of the IDC framework under different tasks. In total, we carry out nine

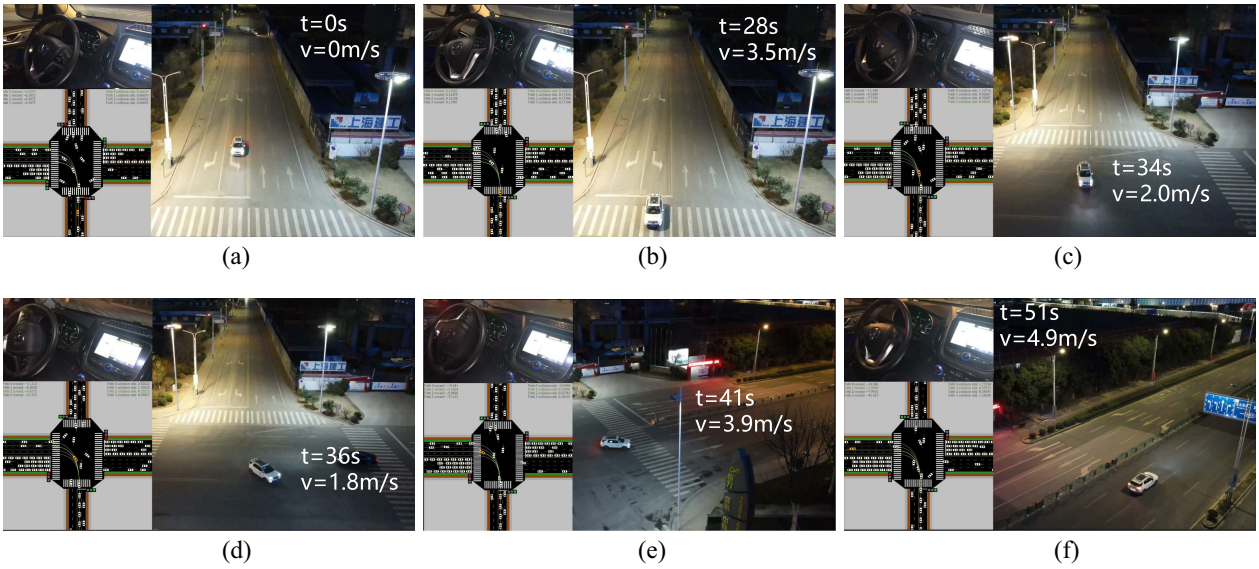


Fig. 9. Featured time steps of the left run. (a) $t = 0$ s, pulling up and waiting for the green signal light. (b) $t = 28$ s, accelerating into the intersection. (c) $t = 34$, slowing down to avoid collision and switching path. (d) $t = 36$ s, accelerating to pass first when selecting a safer path. (e) $t = 41$ s, fixing the path and tracking it. (f) $t = 51$ s, passing the intersection successfully.

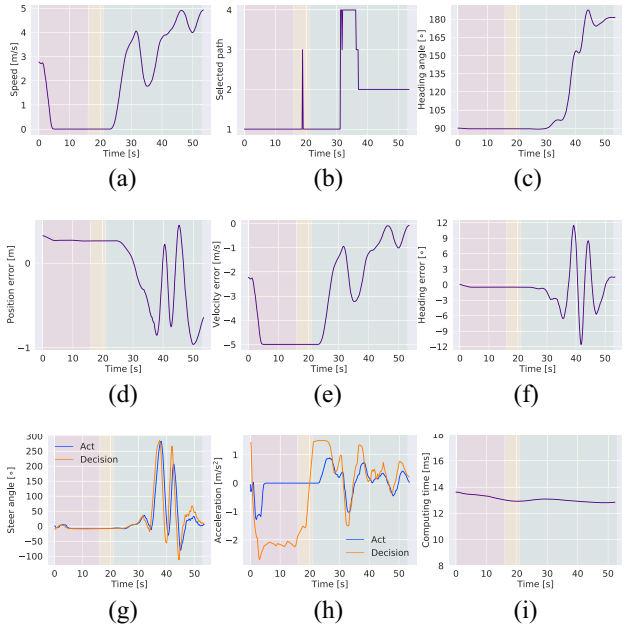


Fig. 10. Key parameters in the left run. (a) Speed. (b) Ref index. (c) Heading angle. (d) Position error. (e) Velocity error. (f) Heading error. (g) Steering angle. (h) Acceleration. (i) Computing time.

runs, three for each task. The ego vehicle and the surrounding vehicles are initialized with random states in each run. Following the diagram in Fig. 8, the run keeps going on until the ego passes the intersection without colliding with obstacles or breaking traffic rules. The diversity among different runs is guaranteed by using different random seeds. All the videos are available online (https://youtu.be/adqjor5KXxQ).

We visualize one of the left runs by snapshotting its featured time steps shown in Fig. 9 and drawing the critical parameters in Fig. 10. In the beginning, the ego pulls up before the stop line, waiting for the green light [Fig. 9(a)]. When that comes, the ego accelerates into the intersection to reduce the velocity tracking error [Fig. 9(b)]. In the center of the intersection, it encounters a straight-go vehicle with high speed from the opposite direction. In order to avoid the collision, the ego slows itself down and switches to path 4, with which it can bypass the vehicle from the back [Fig. 9(c)]. However, another straight-go vehicle comes over after the previous one passes through, but with a relatively low speed. This time, the ego chooses to accelerate to pass first. Interestingly, as the vehicle approaches, the optimal path is automatically selected away from it, i.e., changing from path 4 to path 3 and finally the path 2, to minimize the tracking errors [Fig. 9(c) and (d)]. Following path 2, the ego finally passes the intersection successfully. The computing time of all steps is within 15 ms, showing the superiority of our method in terms of online computing efficiency.

TABLE IV
NOISE LEVELS AND THE CORRESPONDING STANDARD DEVIATION

Noise level	0	1	2	3	4	5	6
δ_p [m]	0	0.017	0.033	0.051	0.068	0.085	0.102
δ_ϕ [°]	0	0.017	0.033	0.051	0.068	0.085	0.102
p_x^j, p_y^j [m]	0	0.05	0.10	0.15	0.20	0.25	0.30
v_{lon}^j [m/s]	0	0.05	0.10	0.15	0.20	0.25	0.30
ϕ^j [°]	0	1.4	2.8	4.2	5.6	7.0	8.4

C. Experiment 2: Robustness to Noise

This experiment aims to compare the driving performance under different noise levels added manually to verify the robustness of the trained policies. Referring to [46], we take a similar measure to divide the noises into seven levels, i.e., 0–6, where all the noises are in the form of Gaussian white noise with different variances varying with the level and are applied in several dimensions of RL states, as shown in Table IV.

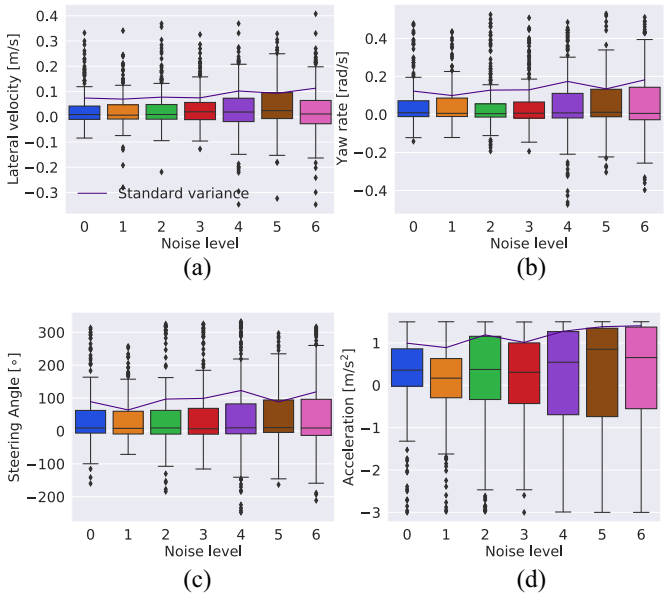


Fig. 11. States and actions in different noise levels. (a) Lateral velocity. (b) Yaw rate. (c) Steering angle. (d) Acceleration.

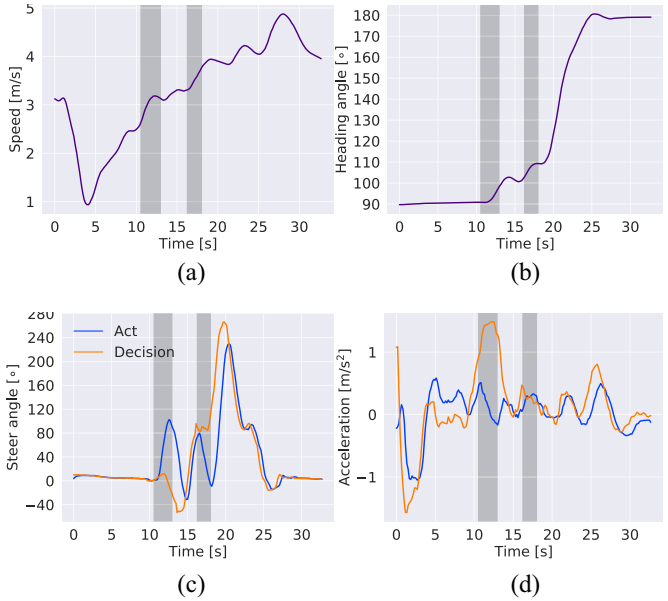


Fig. 12. States and actions under human disturbance. (a) Speed. (b) Heading angle. (c) Steering angle. (d) Acceleration.

We choose the left-turn task to perform seven experiments. For each noise level, i.e., each experiment, we make statistical analysis on the parameters related to vehicle stability, namely, the yaw rate and lateral speed, and the control quantities, i.e., steering angle and acceleration, as shown in Fig. 11. Our method is rather robust to low-level noises (0–3) in which the distribution parameters, including the median value, the standard variance, the quantile values, and the bounds, have no significant change. However, these parameters, especially the variance and the bounds, are inevitably enlarged if we add stronger noise. The fluctuations of the lateral velocity and the yaw rate are mainly caused by the sensitivity of the steering wheel because large noises tend to yield a large variance of the steering angle, which further leads to the swing of the vehicle body. Nevertheless, the stability bounds always remain in a reasonable range, proving the robustness of the proposed method.

D. Experiment 3: Robustness to Human Disturbance

The experiment is to verify the ability of our method to cope with human disturbance. We also use a left-turn case, in which we perform two times of human interventions on the steering wheel. We draw the states of the ego vehicle in Fig. 12, where the colored region is when the human disturbance is acted on. The first one is acted on 10 s when the ego just enters the crossroad, and we turn the steering wheel left to 100° from 0° to make the ego head to the left. After that, the driving system immediately turns the steering wheel right to correct the excessive ego heading. The second one happens at 16 s when the ego is turning to the left to pass the crossroad. We turn the steering wheel right from 90° to 0° to interrupt the process. After the takeover, the driving system is able to turn the steering wheel left to 240° right away to continue to complete the turn left operation. Results show that the proposed method is capable of dealing with the abrupt human disturbance on the steering wheel by quick responses to the interrupted state after taking over.

VII. Conclusion

In this article, we propose the IDC framework for automated vehicles to build an interpretable learning system with high online computing efficiency and applicability among different driving tasks and scenarios. The framework decomposes the driving task into static path planning and dynamic optimal tracking hierarchically. The former generates multiple paths considering static constraints, which are then sent to the latter to be selected and tracked. The latter first formulates the selecting and tracking problem as constrained OCPs mathematically to consider dynamic obstacles. It then solves the problem offline by a model-based RL algorithm to seek an approximate solution in the form of NNs. Notably, these solved approximation functions, namely, value and policy, have a natural correspondence to the selecting and tracking problems, which originates the interpretability. Finally, the value and policy functions are used online instead, releasing the heavy computation in online optimizations. We verify our framework in both simulations and a real-world intersection. Results show that our method owns an order of magnitude higher online computing efficiency compared with the traditional rule-based method. In addition, it yields better driving performance in terms of traffic efficiency and safety and shows great interpretability and adaptability among different driving tasks. About the future work, we will extend the IDC framework to more driving scenarios, including roundabouts, ramps, and realistic urban roads, to further verify its effectiveness. Besides, a suitable state representation will be developed to improve the lower layer's tracking ability and solution precision among paths in different scenarios.

REFERENCES

[1] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, Mar. 2016.

[2] K. K. Santhosh, D. P. Dogra, P. P. Roy, and B. B. Chaudhuri, "Trajectory-based scene understanding using Dirichlet process mixture model," *IEEE Trans. Cybern.*, vol. 51, no. 8, pp. 4148–4161, Aug. 2021.

[3] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *ROBOMECH J.*, vol. 1, no. 1, pp. 1–14, 2014.

[4] G. Li, S. E. Li, B. Cheng, and P. Green, "Estimation of driving style in naturalistic highway traffic using maneuver transition probabilities," *Transp. Res. C, Emerg. Technol.*, vol. 74, pp. 113–125, Jan. 2017.

[5] L. Sun, W. Zhan, and M. Tomizuka, "Probabilistic prediction of interactive driving behavior via hierarchical inverse reinforcement learning," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, 2018, pp. 2111–2117.

[6] L. Hou, L. Xin, S. E. Li, B. Cheng, and W. Wang, "Interactive trajectory prediction of surrounding road users for autonomous driving using structural-LSTM network," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 11, pp. 4615–4625, Nov. 2020.

[7] X. Zhang, J. Ma, Z. Cheng, S. Huang, S. S. Ge, and T. H. Lee, "Trajectory generation by chance-constrained nonlinear MPC with probabilistic prediction," *IEEE Trans. Cybern.*, vol. 51, no. 7, pp. 3616–3629, Jul. 2021.

[8] M. Montemerlo *et al.*, "Junior: The Stanford entry in the urban challenge," *J. Field Robot.*, vol. 25, no. 9, pp. 569–597, 2008.

[9] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 3, pp. 660–674, May/Jun. 1991.

[10] J. Nilsson, Y. Gao, A. Carvalho, and F. Borrelli, "Manoeuvre generation and control for automated highway driving," *IFAC Proc. Vol.*, vol. 47, no. 3, pp. 6301–6306, 2014.

[11] R. Chai, A. Savvaris, A. Tsourdos, S. Chai, and Y. Xia, "Trajectory optimization of space maneuver vehicle using a hybrid optimal control solver," *IEEE Trans. Cybern.*, vol. 49, no. 2, pp. 467–480, Feb. 2019.

[12] R. Chai, A. Savvaris, A. Tsourdos, S. Chai, Y. Xia, and S. Wang, "Solving trajectory optimization problems in the presence of probabilistic constraints," *IEEE Trans. Cybern.*, vol. 50, no. 10, pp. 4332–4345, Oct. 2020.

[13] S. Zhang, W. Deng, Q. Zhao, H. Sun, and B. Litkouhi, "Dynamic trajectory planning for vehicle autonomous driving," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, 2013, pp. 4161–4166.

[14] U. Lee, S. Yoon, H. Shim, P. Vasseur, and C. Demonceaux, "Local path planning in a complex environment for self-driving car," in *Proc. 4th Annu. IEEE Int. Conf. Cyber Technol. Autom. Control Intell.*, 2014, pp. 445–450.

[15] M. K. Ardakani and M. Tavana, "A decremental approach with the A* algorithm for speeding-up the optimization process in dynamic shortest path problems," *Measurement*, vol. 60, pp. 299–307, Jan. 2015.

[16] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," in *Robotics Science and Systems VI*, vol. 104. Cambridge, MA, USA: MIT Press, 2010.

[17] L. Xin *et al.*, "Enable faster and smoother spatio-temporal trajectory planning for autonomous vehicles in constrained dynamic environment," *Proc. Inst. Mech. Eng. D, J. Automobile Eng.*, vol. 235, no. 4, pp. 1101–1112, 2021.

[18] S. E. Li, K. Li, and J. Wang, "Economy-oriented vehicle adaptive cruise control with coordinating multiple objectives function," *Veh. Syst. Dyn.*, vol. 51, no. 1, pp. 1–17, 2013.

[19] S. Li, K. Li, R. Rajamani, and J. Wang, "Model predictive multi-objective vehicular adaptive cruise control," *IEEE Trans. Control Syst. Technol.*, vol. 19, no. 3, pp. 556–566, May 2011.

[20] S. Li, "Reinforcement learning for decision and control," Lecture Notes, Tsinghua Univ., Beijing, China, 2020. [Online]. Available: http://www.idlab-tsinghua.com/thulab/labweb/publications.html

[21] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3826–3839, Sep. 2020.

[22] K. Li, T. Zhang, and R. Wang, "Deep reinforcement learning for multiobjective optimization," *IEEE Trans. Cybern.*, vol. 51, no. 6, pp. 3103–3114, Jun. 2021.

[23] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electron. Imag.*, vol. 29, no. 19, pp. 70–76, 2017.

[24] P. Wang, C.-Y. Chan, and A. de La Fortelle, "A reinforcement learning based approach for automated lane change maneuvers," in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2018, pp. 1379–1384.

[25] D. C. K. Ngai and N. H. C. Yung, "A multiple-goal reinforcement learning method for complex vehicle overtaking maneuvers," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 2, pp. 509–522, Jun. 2011.

[26] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[27] Z. Ni, N. Malla, and X. Zhong, "Prioritizing useful experience replay for heuristic dynamic programming-based learning systems," *IEEE Trans. Cybern.*, vol. 49, no. 11, pp. 3911–3922, Nov. 2019.

[28] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. 4th Int. Conf. Learn. Represent. (ICLR)*, San Juan, Puerto Rico, May 2016. [Online]. Available: https://arxiv.org/abs/1509.02971

[29] J. Duan, S. E. Li, Y. Guan, Q. Sun, and B. Cheng, "Hierarchical reinforcement learning for self-driving decision-making without reliance on labelled driving data," *IET Intell. Transp. Syst.*, vol. 14, no. 5, pp. 297–305, 2020.

[30] Y. Guan, Y. Ren, S. E. Li, Q. Sun, L. Luo, and K. Li, "Centralized cooperation for connected and automated vehicles at intersections by proximal policy optimization," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 12597–12608, Nov. 2020.

[31] J. Chen, B. Yuan, and M. Tomizuka, "Model-free deep reinforcement learning for urban autonomous driving," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, 2019, pp. 2765–2771.

[32] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artif. Intell.*, vol. 267, pp. 1–38, Feb. 2019.

[33] B. Kim, R. Khanna, and O. O. Koyejo, "Examples are not enough, learn to criticize! criticism for interpretability," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 2288–2296.

[34] Q. Ge, Q. Sun, S. E. Li, S. Zheng, W. Wu, and X. Chen, "Numerically stable dynamic bicycle model for discrete-time control," in *Proc. IEEE Intell. Veh. Symp. Workshops (IV Workshops)*, 2021, pp. 128–134.

[35] X. Yang, H. He, and X. Zhong, "Approximate dynamic programming for nonlinear-constrained optimizations," *IEEE Trans. Cybern.*, vol. 51, no. 5, pp. 2419–2432, May 2021.

[36] B. Karg and S. Lucia, "Efficient representation and approximation of model predictive control laws via deep learning," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3866–3878, Sep. 2020.

[37] Z. Allen-Zhu, Y. Li, and Z. Song, "A convergence theory for deep learning via over-parameterization," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 242–252.

[38] P. A. Lopez *et al.*, "Microscopic traffic simulation using SUMO," in *Proc. 21st IEEE Int. Conf. Intell. Transp. Syst.*, 2018, pp. 2575–2582. [Online]. Available: https://elib.dlr.de/124092/

[39] Y. Guan, J. Duan, S. E. Li, J. Li, J. Chen, and B. Cheng, "Mixed policy gradient," 2021, *arXiv:2102.11513*.

[40] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," 2015, *arXiv:1511.07289*.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

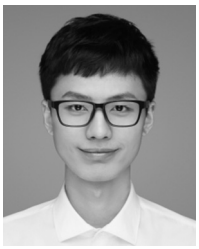
[42] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2009, pp. 1879–1884.

[43] W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha, "A real-time motion planner with trajectory optimization for autonomous vehicles," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2012, pp. 2061–2067.

[44] J. Chen, C. Tang, L. Xin, S. E. Li, and M. Tomizuka, "Continuous decision making for on-road autonomous driving under uncertain and interactive environments," in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2018, pp. 1651–1658.

[45] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.

[46] J. Duan, "Study on distributional reinforcement learning for decision-making in autonoumous driving," Ph.D. dissertation, School. Vehicle and Mobility, Tsinghua Univ., Beijing, China, 2021.



Yang Guan received the B.S. degree from the School of Mechanical Engineering, Beijing Institute of Technology, Beijing, China, in 2017. He is currently pursuing the Ph.D. degree with the School of Vehicle and Mobility, Tsinghua University, Beijing.

His research interests include decision making of autonomous vehicle and reinforcement learning.

Mr. Guan was a recipient of the Best Student Paper Award of IEEE International Conference on Intelligent Transportation Systems in 2020 and 2021.

Yangang Ren received the B.S. degree from the Department of Automotive Engineering, Tsinghua University, Beijing, China, in 2018, where he is currently pursuing the Ph.D. degree with the School of Vehicle and Mobility.

His research interests include decision and control of autonomous driving, reinforcement learning, and adversarial learning.

Mr. Ren was a recipient of the Best Student Paper Award of IEEE International Conference on Intelligent Transportation Systems in 2020 and 2021.

Qi Sun received the Ph.D. degree in automotive engineering from the Ecole Centrale de Lille, Lille, France, in 2017.

He did scientific research and completed his Ph.D. dissertation with the CRIStAL Research Center, Ecole Centrale de Lille from 2013 to 2016. He is currently a Postdoctoral Fellow with the State Key Laboratory of Automotive Safety and Energy and the School of Vehicle and Mobility, Tsinghua University, Beijing, China. His active research interests include intelligent vehicles, automatic driving technology, distributed control, and optimal control.

Shengbo Eben Li (Senior Member, IEEE) received the M.S. and Ph.D. degrees from Tsinghua University, Beijing, China, in 2006 and 2009, respectively.

He worked with Stanford University, Stanford, CA, USA, the University of Michigan at Ann Arbor, Ann Arbor, MI, USA, and the University of California at Berkeley, Berkeley, CA, USA. He is currently a Tenured Professor with Tsinghua University. He is the author of over 100 journal articles/conference papers and the co-inventor of over 20 Chinese patents. His active research interests include intelligent vehicles and driver assistance, reinforcement learning and distributed control, and optimal control and estimation.

Dr. Li was a recipient of the Best Paper Award at the 2014 IEEE ITS Symposium, the Best Paper Award at the 14th ITS Asia–Pacific Forum, the National Award for Technological Invention in China in 2013, the Excellent Young Scholar of NSF China in 2016, and the Young Professorship of Changjiang Scholar Program in 2016. He serves as an Associate Editor for *IEEE Intelligent Transportation Systems Magazine* and IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS.

Haitong Ma (Graduate Student Member, IEEE) received the B.S. degree in automotive engineering from Tsinghua University, Beijing, China, in 2019, where he is currently pursuing the M.S. degree with the School of Vehicle and Mobility, under the guidance of Prof. S. Zheng and Prof. S. E. Li.

His research interests include optimal control and safe control.

Jingliang Duan received the B.S. degree from the College of Automotive Engineering, Jilin University, Changchun, China, in 2015, and the Ph.D. degree from the School of Vehicle and Mobility, Tsinghua University, Beijing, China, in 2021.

He was a Visiting Student Researcher with the Department of Mechanical Engineering, University of California at Berkeley, Berkeley, CA, USA, in 2019. His research interests include decision and control of autonomous vehicles, reinforcement learning and adaptive dynamic programming, and driver behavior analysis.

Yifan Dai received the Ph.D. degree in automotive engineering from Tsinghua University, Beijing, China, in 2013.

He is currently the Assistant Dean of Tsinghua University–Suzhou Automotive Research Institute and the Director of Jiangsu Intelligent and Connected Automotive Innovation Center. His active research interests include autonomous vehicles and vehicle dynamics control.

Bo Cheng received the B.S. and M.S. degrees in automotive engineering from Tsinghua University, Beijing, China, in 1985 and 1988, respectively, and the Ph.D. degree in mechanical engineering from The University of Tokyo, Tokyo, Japan, in 1998.

He is currently a Professor with the School of Vehicle and Mobility, Tsinghua University, and the Dean of Tsinghua University–Suzhou Automotive Research Institute. He is the author of more than 100 peer-reviewed journal articles/conference papers and the co-inventor of 40 patents. His active research interests include autonomous vehicles, driver-assistance systems, active safety, and vehicular ergonomics.

Dr. Cheng is also the Chairman of the Academic Board of SAE-Beijing, a member of the Council of the Chinese Ergonomics Society, and a Committee Member of the National 863 Plan.