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

Yang Guan[1], Yangang Ren[1], Qi Sun[1], Shengbo Eben Li*[1], Haitong Ma[1], Jingliang Duan[1], Yifan Dai[2], Bo Cheng[1]

***Abstract*—Decision and control are core functionalities of high-level automated vehicles. Current mainstream methods, such as functionality decomposition and end-to-end reinforcement learning (RL), either suffer high time complexity or poor interpretability and adaptability on real-world autonomous driving tasks. In this paper, 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 reinforcement learning (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 tasks. The effectiveness of the proposed method is also demonstrated in real road tests with complicated traffic conditions.**

***Index Terms*—Automated vehicle, Decision and control, Reinforcement learning, Model-based.**

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 the road transportation. Decision and control are indispensable for high-level autonomous driving, which are 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: decomposed scheme and end-to-end scheme.

Decomposed scheme splits the decision and control functionality into several serial submodules, such as prediction, behavior selection, trajectory planning and control [1]. Prediction is to predict the future trajectory of traffic participants to determine the feasible region in future time steps [2]. It is further decomposed into behavior recognition [3], [4] and trajectory prediction [5], [6]. Since the prediction algorithms usually works on each surrounding vehicle, it means that the more the number of vehicles, the more computation is needed. Behavior selection is then used to choose a high-level driving behavior relying on an expert system in which many designed rules are embedded. Typical methods include finite state machine [7] and decision tree [8]. Based on the selected behavior, a collision free space-time curve satisfying vehicle dynamics is calculated according to the predicted trajectories and road constraints by the trajectory planning submodule. Three main categories of the planning algorithms include optimization-based, search-based and sample-based. Optimization-based methods formulate the planning problem into an optimization problem, where specific aspects of trajectory are optimized and constraints are considered [9], [10]. However, it suffers from long computational time for large-scale nonlinear and non-convex problem. The search-based methods represented by A* and Rapidly-exploring Random Tree are more efficient [11]–[16], but they usually lead to low-resolution paths and can barely take dynamic obstacles into consideration. The sample-based methods also have poor computing efficiency because they sample points and interpolate them evenly in the whole state space [17], [18]. Xin *et al.* proposed a combination of the optimization-based and search-based methods, where a trajectory is searched in the space-time by A* and then smoothed by model predictive control (MPC), yielding the best performance in terms of planning time and comfort [19]. Finally, the controller is used to follow the planned trajectory and calculate the expected controls by linear quadratic regulator or MPC [20], [21]. The decomposed scheme requires large amount of human design but is still hard to cover all possible driving scenarios due to the long tail effect. Besides, the real time performance cannot be guaranteed because it is time-consuming to complete all the works serially in a limited time for industrial computers.

End-to-end scheme computes the expected instructions directly from inputs given by perception module using a policy usually carried out by a deep neural network (NN). Rein-

This work is supported by International Science & Technology Cooperation Program of China under 2019YFE0100200, NSF China with 51575293, and U20A20334. It is also partially supported by Tsinghua University-Toyota Joint Research Center for AI Technology of Automated Vehicle.

[1]School of Vehicle and Mobility, Tsinghua University, Beijing, 100084, China. [2]Suzhou Automotive Research Institute, Tsinghua University, Suzhou, 215200, China. All correspondence should be sent to S. Eben Li. <lisb04@gmail.com>.

forcement learning (RL) methods do not rely on labelled driving data but learn by trial-and-error in real-world or a high fidelity simulator [22], [23]. Early RL applications on autonomous driving mainly focus on learning a single driving behavior, e.g., lane keeping [24], lane changing [25] or overtaking [26]. They usually employ deep Q-networks [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.* 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.* achieved centralized control in a four-leg single-lane intersection with sparse rewards, in which only eight cars are considered [29]. Chen *et al.* 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 [30]. 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 travelled towards a destination, collisions with other road users or scene objects, maintaining comfort and stability while avoiding extreme acceleration, braking or steering. It is non-trivial and needs a lot of human efforts 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.

In this paper, we propose an integrated decision and control framework (IDC) 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:

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, 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. The IDC framework is computationally efficient because we unload the heavy online optimizations by solving the constrained OCPs offline in the form of value and policy NNs using RL for path selecting and tracking thereafter. It is interpretable in the sense that the solved value and policy functions are the approximation for the optimal cost and the optimal action of the constrained OCP. Moreover, the IDC employs RL to solve a task-independent OCP with tracking errors as objective and safety constraints, making it applicable among a variety of scenarios and tasks.

2) We develop a model-based RL algorithm called generalized exterior point method (GEP) for the purpose of approximately solving OCP with large-scale state-wise constraints. The GEP is in fact an extension of the exterior point method in the optimization domain to the field of NN, in which it first constructs an extensive problem involving all the candidate paths and transforms it into an unconstrained problem with a penalty on safety violations. Afterward, the approximate feasible optimal control policy is obtained by alternatively performing gradient descent and enlarging the penalty. The convergence of the GEP is proved. The GEP is the core of IDC because it can deal with a large number of state-wise constraints explicitly and update NNs efficiently with the guidance of model. To the best of our knowledge, GEP is the first model-based solver for OCPs with large-scale state-wise constraints that are parameterized by NNs.

3) We evaluate the proposed method extensively in both simulations and in a real-world road to verify the performance in terms of online computing efficiency, safety, and task adaptation, etc. The results show the potential of the method to be applied in real-world autonomous driving tasks.

II. Integrated Decision and Control Framework

In this section, we introduce the framework of integrated decision and control framework (IDC). As shown in Fig. 1, the framework consists of two layers: static path planning and dynamic optimal tracking.

Different from 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.

The lower layer further considers the candidate paths and the dynamic information such as surrounding vehicles, pedestrians and bicycles. For each candidate path, a constrained optimal control problem (OCP) is formulated and optimized to choose the optimal path and find the control command. The objective function is to minimize the tracking error within a finite horizon and the constraints characterize safety requirements. In each time step, the optimal path is chosen as the one with the lowest optimal cost function and thereafter tracked. The core of our method is to substitute all the expensive online optimizations with feed-forward of two neural networks (NNs) trained offline by reinforcement learning (RL). Specifically, 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 that is capable of tracking different shape of paths while maintaining the ability to avoid collisions. Meanwhile, a value NN called critic is learned to approximate the optimal cost of tracking different paths, for the purpose of 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 to embed key parameters of paths

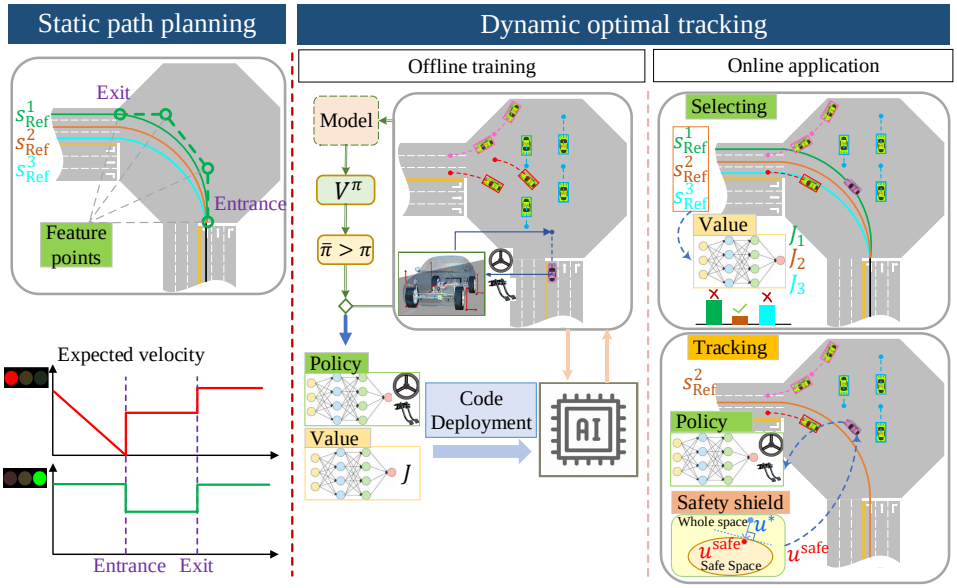


Fig. 1: Illustration of the integrated decision and control framework.

planned in advance into the electronic map and read from it directly. 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, it can be easily transferred among different driving scenarios without a lot of human design. As the upper layer only uses the 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, which all can be solved by the developed model-based solver, saving time to design separate reward functions for different tasks. Third, the IDC framework are interpretable in the way that the learned value and policy function approximate the optimal value and the optimal action of the constrained OCPs. These optimal solutions can also be obtained by model predictive control (MPC) to verify the correctness of the trained NNs. Besides, the functionality partitioning in the IDC framework helps us to explain the results of path selection and tracking accordingly.

III. Static Path Planning

This module aims to generate multiple candidate paths for optimal tracking of the lower layer meanwhile maintaining high computational efficiency. For that purpose, the Cubic Bezier curve is adopted to obtain a continuous and smooth path. Given the road map, we choose four feature points to shape of Bezier curve and generate multiple paths with considerations of static traffic information such as road topology, traffic rules, etc., following the Algorithm 1. We summary two strategies to generate candidate paths. One is the pre-generating method, in which these paths are produced in advance and do not change with the riding of the ego vehicle, as illustrated in Fig. 1. The other one is the real-time generating method where the paths are planned in real-time and always start from the ego vehicle. Apparently, the former is simple and convenient to be directly embedded in the map, and has higher efficiency because it only needs to read the pre-stored paths every step. By contrast, the latter has higher flexibility and may conduct more complex driving behaviors, but its online computing efficiency will be a bit lower than the former. Both methods will be much more efficient than current existing methods as they simply generate regular candidate paths without considering collision avoidance with the dynamic obstacles. Note that the planned paths only serve as references for the lower layer, the actual travelled path can be largely different due to further considerations on dynamic obstacles.

Algorithm 1 Static path planning

Initialize: discrete point number M
for each path **do**
  Choose four feature points: start point (X_0, Y_0), control points (X_1, Y_1),(X_2, Y_2), and end point (X_3, Y_3)
  for $t = 0 : 1/M : 1$ **do**
    $p_{\mathrm{x}}^{\mathrm{ref}}(t) = X_0(1-t)^3 + 3X_1t(1-t)^2 + 3X_2t^2(1-t) + X_3t^3$
    $p_{\mathrm{y}}^{\mathrm{ref}}(t) = Y_0(1-t)^3 + 3Y_1t(1-t)^2 + 3Y_2t^2(1-t) + Y_3t^3$
    $\phi^{\mathrm{ref}}(t) = \arctan(\frac{Y(t)-Y(t-1)}{X(t)-X(t-1)})$
  end for
  Output $\{(p_{\mathrm{x}}^{\mathrm{ref}}, p_{\mathrm{y}}^{\mathrm{ref}}, \phi^{\mathrm{ref}})\}$
end for

As for the expected velocity, we heuristically assign different speed levels with respect to road regions, traffic signals as well as 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. Similar to the candidate paths, the expected velocity provides a goal for the lower layer to track but not necessarily to follow strictly, so that the ego vehicle seeks to minimize the tracking error while satisfying safety constraints. Actually, it can be simply a fixed value, 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

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 certain 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:

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

world model

where 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 j-th vehicle in the interested vehicle set I. Q, R, M are positive-definite weighting matrices. F_{ego} represents the bicycle vehicle dynamics with linear tire model. F_{pred}, on the other hand, is the surrounding vehicle prediction model. 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 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:

$$\begin{aligned}
x_{i|t}^{\text{ref}} &= \begin{bmatrix} p_{\text{x}}^{\text{ref}} \\ p_{\text{y}}^{\text{ref}} \\ v_{\text{lon}}^{\text{ref}} \\ 0 \\ \phi^{\text{ref}} \\ 0 \end{bmatrix}_{i|t} \quad x_{i|t}^{\text{road}} = \begin{bmatrix} p_{\text{x}}^{\text{road}} \\ p_{\text{y}}^{\text{road}} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}_{i|t} \quad x_{i|t} = \begin{bmatrix} p_{\text{x}} \\ p_{\text{y}} \\ v_{\text{lon}} \\ v_{\text{lat}} \\ \phi \\ \omega \end{bmatrix}_{i|t} \\
x_{i|t}^j &= \begin{bmatrix} p_{\text{x}}^j \\ p_{\text{y}}^j \\ v_{\text{lon}}^j \\ 0 \\ \phi^j \\ 0 \end{bmatrix}_{i|t} \quad u_{i|t} = \begin{bmatrix} \delta \\ a \end{bmatrix}_{i|t} \quad M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}
\end{aligned} \tag{2}$$

$$F_{\text{ego}} = \begin{bmatrix} p_{\text{x}} + \Delta t (v_{\text{lon}} \cos\phi - v_{\text{lat}} \sin\phi) \\ p_{\text{y}} + \Delta t (v_{\text{lon}} \sin\phi + v_{\text{lat}} \cos\phi) \\ v_{\text{lon}} + \Delta t (a + v_{\text{lat}} \omega) \\ \frac{m v_{\text{lon}} v_{\text{lat}} + \Delta t [(L_f k_f - L_r k_r)\omega - k_f \delta v_{\text{lon}} - m v_{\text{lon}}^2 \omega]}{m v_{\text{lon}} - \Delta t (k_f + k_r)} \\ \phi + \Delta t \omega \\ \frac{-I_z \omega v_{\text{lon}} - \Delta t [(L_f k_f - L_r k_r) v_{\text{y}} - L_f k_f \delta v_{\text{lon}}]}{\Delta t (L_f^2 k_f + L_r^2 k_r) - I_z v_{\text{lon}}} \end{bmatrix} \tag{3}$$

$$F_{\text{pred}} = \begin{bmatrix} p_{\text{x}}^j + \Delta t (v_{\text{lon}}^j \cos\phi^j - v_{\text{lat}}^j \sin\phi^j) \\ p_{\text{y}}^j + \Delta t (v_{\text{lon}}^j \sin\phi^j + v_{\text{lat}}^j \cos\phi^j) \\ v_{\text{lon}}^j \\ 0 \\ \phi^j + \Delta t \omega_{\text{pred}}^j \\ 0 \end{bmatrix} \tag{4}$$

where $p_{\text{x}}, p_{\text{y}}$ are the position coordinates, for ego and other vehicles, it is the position of their respective center of gravity (CG), $v_{\text{lon}}, v_{\text{lat}}$ are the longitudinal and lateral velocities, ϕ is the heading angle, ω is the yaw rate, δ and a are the front wheel angle and the acceleration commands, respectively. The ego dynamics model is discretized by the first-order Euler method, which has been proved to be numerically stable at any low speed [31]. Other vehicle parameters are listed in Table I. The vehicle prediction model is a simple deduction from the current state with constant speed and turning rate w_{pred}^j, which depends on the driving scenarios and will be specified in the experiments.

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

$$\tau^* = \arg\min_{\tau} \{ J_\tau^* | \tau \in \Pi \} \tag{5}$$

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 value J_τ^*. Such a criterion of path selection is consistent with the objective of the optimal control problem (1) in the sense that it optimizes the problem with respect to paths within the candidate path set, compared to the original optimization with respect to control quantities.

In such framework of selecting and tracking, the lower layer is able to well determine a control quantity that yields good driving efficiency and, meanwhile, the safety guarantee. But unfortunately, so far, the time complexity is extremely high for on-board vehicle computing devices, as in each time step we need to solve N constrained optimal control problem, while each of them owns up to hundreds of variables and thousands of constrains. 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, the computation burden in the lower layer can be almost nearly eliminated.

B. Offline training

1) Complete RL problem formulation: We aim to solve the path selecting and path tracking problems (1) and (5) by RL. Notice that there are several significant differences between the OCP (1) and RL problems, originated from the online or offline optimizations. The OCP, which is designed for online optimization, seeks to find a single optimal control quantity of a single state given a specific path at time step t. RL problems, on the other hand, aim to solve a parameterized control policy called actor that maps from state space $\mathcal{S}$ to control space $\mathcal{A}$ as well as a parameterized value function called critic that evaluates the preference to a state, in an offline style. Therefore, in RL problems the variable to be optimized is no longer the control quantity but the parameters of the actor and critic that usually in the form of NN. In addition, the objective function and constraints of RL are not about a single state any more, but about a state distribution in the state space. The state $s \in \mathcal{S}$ is the input of the actor and critic, which is designed to contain necessary information to determine driving actions. Except for the information of the ego vehicle and surrounding vehicles, we also incorporate the path information as a part of states to get a policy that can handle tracking tasks for different paths. The resulting RL problem is shown as:

$$\begin{aligned} \min_{\theta} \quad & J_{\text{actor}} = \mathbb{E}_{s_{0|t}}\left\{\sum_{i=0}^{T-1} l(s_{i|t}, \pi_\theta(s_{i|t}))\right\} \\ \text{s.t.} \quad & s_{i+1|t} = f(s_{i|t}, \pi_\theta(s_{i|t})) \\ & g_e(s_{i|t}) \geq 0, e \in E \\ & s_{0|t} = s_t \leftarrow \{\tau, x_t, x_t^j, j \in J\} \sim d, \\ & i = 0 : T-1 \end{aligned} \tag{6}$$

$$\begin{aligned} \min_{w} \quad & J_{\text{critic}} = \mathbb{E}_{s_{0|t}}\left\{\left(\sum_{i=0}^{T-1} l(s_{i|t}, \pi_\theta(s_{i|t})) - V_w(s_{0|t})\right)^2\right\} \\ \text{s.t.} \quad & s_{i+1|t} = f(s_{i|t}, \pi_\theta(s_{i|t})) \\ & s_{0|t} = s_t \sim d, \\ & i = 0 : T-1 \end{aligned} \tag{7}$$

where $s_t \leftarrow \{\tau, x_t, x_t^j, j \in J\}$ denotes the state is constructed using the information of reference path, ego vehicle state and surrounding vehicle states. $l(s_{i|t}, \pi_\theta(s_{i|t})) := (x_{i|t}^{\text{ref}} - x_{i|t})^\top Q (x_{i|t}^{\text{ref}} - x_{i|t}) + \pi_\theta^\top(s_{i|t}) R \pi_\theta(s_{i|t})$. f denotes the system model, which 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. d denotes the state distributions sampled from the environment. $\pi_\theta : \mathcal{S} \to \mathcal{A}$ and $V_w : \mathcal{S} \to \mathbb{R}$ are actor and critic, parameterized by θ and w that are generally in form of NNs, respectively. From the results of [32], given over-parameterized NNs, the optimal policy π_{θ_*} of (6) 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 (7) from s_t, i.e.,

$$\begin{aligned} u_t^* &= \pi_{\theta_*}(s_t), \ \forall s_t \in \mathcal{S} \\ J^* &= V_{w_*}(s_t), \ \forall s_t \in \mathcal{S} \end{aligned} \tag{8}$$

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.

2) Solver - GEP: To solve the converted RL problem, we adopt the policy iteration framework, wherein two procedures, namely policy evaluation and policy improvement, are alternatively performed to update the critic and 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 infinite number of state constraints. To tackle this, we propose a model-based RL algorithm called generalized exterior point method (GEP) adapted from the one in the optimization field. It first transforms the constrained problem (6) into an unconstrained one by the exterior penalty function, shown as:

$$\begin{aligned} \min_{\theta} \quad & J_p = J_{\text{actor}} + \rho J_{\text{penalty}} \\ & = \mathbb{E}_{s_{0|t}}\left\{\sum_{i=0}^{T-1} l(s_{i|t}, \pi_\theta(s_{i|t}))\right\} + \rho \mathbb{E}_{s_{0|t}}\left\{\sum_{i=0}^{T-1} \varphi_i(\theta)\right\} \\ \text{s.t.} \quad & s_{i+1|t} = f(s_{i|t}, \pi_\theta(s_{i|t})) \\ & \varphi_i(\theta) = \sum_{e \in E} [\max\{0, -g_e(s_{i|t})\}]^2 \\ & s_{0|t} = s_t \sim d, \\ & i = 0 : T-1 \end{aligned} \tag{9}$$

where φ is the penalty function, 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 exterior point method, the optimizing procedure of GEP does not necessarily find the optimal solution of the unconstrained problem (9), but we will prove that GEP still converges to the optimal policy under certain conditions. It can be seen that GEP is simple to implement and is powerful to deal with large-scale parameter space facilitated by the gradient descent technique. Besides, from the form of (9), numerous state constraints can be handled naturally by regarding the constraint violation as a term of utility function multiplied by ρ. The training pipeline is shown in Algorithm 2.

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 (9) 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*

$$J_p(\theta_k, \rho_k) \leq \min_{\theta} J_p(\theta, \rho_k) + \Delta_k, k = 1, 2, \ldots \tag{10}$$

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

Algorithm 2 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 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 non-increasing sequence $\delta_k, k \geq 1$ that satisfies*

$$\begin{aligned} &\Delta_k = \delta_k - \delta_{k+1}, \\ &\delta_k \geq \delta_{k+1}, \lim_{k\to\infty} \delta_k = 0 \end{aligned} \tag{11}$$

Proof. We can construct such a sequence by setting

$$\delta_1 = \sum_{i=1}^{\infty} \Delta_i, \delta_{k+1} = \delta_k - \Delta_k, k \geq 1 \tag{12}$$

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

$$\lim_{k\to\infty} \delta_k = \lim_{k\to\infty} \delta_1 - \sum_{i=1}^{k-1} \Delta_i = \delta_1 - \lim_{k\to\infty} \sum_{i=1}^{k-1} \Delta_i = 0 \tag{13}$$

□

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*

$$J_p(\theta_{k+1}, \rho_{k+1}) - \delta_{k+1} \geq J_p(\theta_k, \rho_k) - \delta_k \tag{14}$$

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

$$\begin{aligned} J_p(\theta_{k+1}, \rho_{k+1}) &= J_{\text{actor}}(\theta_{k+1}) + \rho_{k+1} J_{\text{penalty}}(\theta_{k+1}) \\ &\geq J_{\text{actor}}(\theta_{k+1}) + \rho_k J_{\text{penalty}}(\theta_{k+1}) \\ &= J_p(\theta_{k+1}, \rho_k) \end{aligned} \tag{15}$$

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

$$\begin{aligned} &J_p(\theta_{k+1}, \rho_{k+1}) \geq J_p(\theta_k, \rho_k) - \Delta_k \\ &J_p(\theta_{k+1}, \rho_{k+1}) - \delta_{k+1} \geq J_p(\theta_k, \rho_k) - \delta_k \end{aligned} \tag{16}$$

□

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

$$J_{\text{actor}}(\theta_*) - \delta_{k+1} \geq J_p(\theta_k, \rho_k) - \delta_k \geq J_{\text{actor}}(\theta_k) - \delta_k \tag{17}$$

Proof. Because θ_* is the optimal solution of the problem (6), it has $J_{\text{penalty}}(\theta_*) = 0$. Then from the Assumption 1 the first inequality is obtained,

$$\begin{aligned} &J_{\text{actor}}(\theta_*) = J_p(\theta_*, \rho_k) \geq \min_\theta J_p(\theta, \rho_k) \geq J_p(\theta_k, \rho_k) - \Delta_k \\ &J_{\text{actor}}(\theta_*) - \delta_{k+1} \geq J_p(\theta_k, \rho_k) - \delta_k \end{aligned} \tag{18}$$

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

$$J_p(\theta_k, \rho_k) = J_{\text{actor}}(\theta_k) + \rho_k J_{\text{penalty}}(\theta_k) \geq J_{\text{actor}}(\theta_k) \tag{19}$$

□

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 (6). From Lemma 2 and Lemma 3, we can see that $\{J_p(\theta_{k_j}, \rho_{k_j}) - \delta_{k_j}\}$ is a non-increasing sequence with upper limit J_{actor}^*, therefore

$$\lim_{k_j\to\infty} (J_p(\theta_{k_j}, \rho_{k_j}) - \delta_{k_j}) = \lim_{k_j\to\infty} J_p(\theta_{k_j}, \rho_{k_j}) = J_p^* \leq J_{\text{actor}}^* \tag{20}$$

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

$$\lim_{k_j\to\infty} \rho_{k_j} J_{\text{penalty}}(\theta_{k_j}) = J_p^* - J_{\text{actor}}(\bar{\theta}) \tag{21}$$

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

$$\lim_{k_j\to\infty} J_{\text{penalty}}(\theta_{k_j}) = J_{\text{penalty}}(\bar{\theta}) = 0 \tag{22}$$

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

$$J_{\text{actor}}(\bar{\theta}) \leq J_{\text{actor}}^* \Rightarrow J_{\text{actor}}(\bar{\theta}) = J_{\text{actor}}^* \tag{23}$$

where the equation holds by the definition of J_{actor}^*. Equation (23) indicates the optimality of the limit $\bar{\theta}$. □

C. Online application

Ideally, we expect to get an optimal policy which is able to output the optimal action within the safety action space in any given point in the state space. Unfortunately, it is impossible to acquire such a policy in both theory and practice. First, the converted RL problem (6) enforces constraint on every single state point, leading to an infinite number of constraints in the continuous state space. But nevertheless when we solve the equivalent unconstrained problem (9), we approximate the expectation by an average of samples, that means we only consider finite constraints of the set of sample 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 (8) that the approximation function has infinite fitting power cannot be established in practical, resulting in a suboptimal solution without safety guarantee. To ensure the safety performance, we adopt a multi-step safety shield after the output of the policy.

1) Multi-step 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:

$$u_t^{\text{safe}} = \begin{cases} u_t^*, & \text{if} \quad u_t^* \in \mathcal{U}_{\text{safe}}(s_t) \\ \arg \min\limits_{u \in \mathcal{U}_{\text{safe}}(s_t)} \|u - u_t^*\|_2^2, & \text{else} \end{cases} \tag{24}$$

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,

$$\mathcal{U}_{\text{safe}}(s_t) = \{u_t | g_e(s_{i|t}) \geq 0, i = 1, \ldots, n_{ss}, e \in E\} \tag{25}$$

2) Algorithm for online application: Given the trained policy and value functions, in online application, we simply construct a set of states for different paths, then pass them to the trained value function to select the one with the lowest value, which is next passed to the trained policy to get the optimal control, as summarized in Algorithm 3.

Algorithm 3 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 (24)

Apply u_{safe}^* to observe x_{t+1} and $x_{t+1}^j, j \in I$

end for

V. Simulation Verification

A. Scenario and task descriptions

We first carried out our experiments on a regular signalized four-way intersection built in the simulation, where the roads in different directions are all the six-lane dual carriageway, as shown in Fig. 2. The junction is a square with a side length of 50m. Each entrance of the intersection has three lanes, each with a width of 3.75m, for turning left, going straight and turning right, respectively. With the help of SUMO [33], we generate a dense traffic flow of 800 vehicles per hour on each lane. These vehicles are controlled by the car-following and lane-changing models in the SUMO, producing a variety of traffic behaviors. Moreover, a two-phase traffic signal is included to control the traffic flow of turning left and going straight. We verify our algorithm in three tasks: turn left, go straight and turn right. In each task, the ego vehicle is initialized randomly from the south entrance and is expected to drive safely and efficiently to pass the intersection.

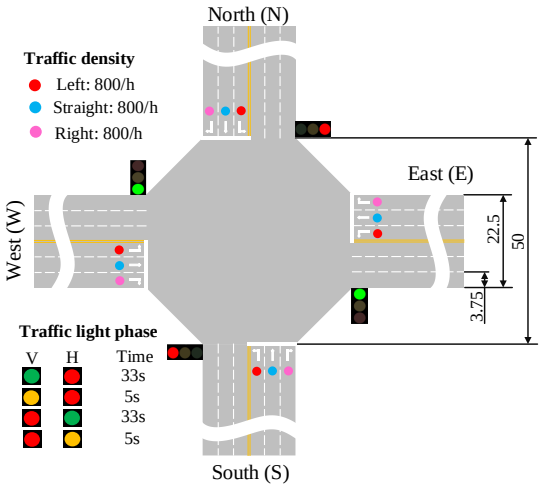


Fig. 2: The scenario used for experiment verification.

B. Implementation of our algorithm

1) Path planning: In this paper, we adopt the static path planning method to generate multiple candidate paths. Specially, in our scenario, each task is assigned three paths according to the lane number of exits. The paths are generated by the cubic Bezier curve featured by four key points. The expected velocity is chosen as a fixed value for simplicity.

2) State and utility function: As mentioned in section IV-B1, the state should be designed to include information of the ego vehicle, the surrounding vehicles and the reference path, i.e.,

$$s_t = [s_t^{\text{ego}}, s_t^{\text{other}}, s_t^{\text{ref}}] \tag{26}$$

where $s_t^{\text{ego}} = x_t$ is the ego dynamics defined in section IV-A, s_t^{other} is the concatenation of the interested surrounding vehicles. Take the turn left task as an example, it is defined as:

$$\begin{aligned} s_t^{\text{other}} &= [p_x^j, p_y^j, \phi^j, v_{\text{lon}}^j]_{t, j \in I_{\text{left}}} \\ I_{\text{left}} &= [\text{SW1}, \text{SW2}, \text{SN1}, \text{SN2}, \text{NS1}, \text{NS2}, \text{NW1}, \text{NW2}] \end{aligned} \tag{27}$$

where I_{left} is an ordered list of vehicles that have potential conflicts with the ego. They are encoded by their respect route start and end, as well as the order on that. Correspondingly, one can define the go straight and turn right task in a similar way. The information of the reference s_t^{ref}, however, is designed in an implicit way by the tracking errors with respect to the position, the heading angle, and the velocity:

$$s_t^{\text{ref}} = [\delta_p, \delta_\phi, \delta_v]_t \tag{28}$$

where δ_p is the position error, $|\delta_p| = \sqrt{(p_x - p_x^{\text{ref}})^2 + (p_y - p_y^{\text{ref}})^2}$, $\text{sign}(\delta_p)$ 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. 3. The weighting matrices in the utility function

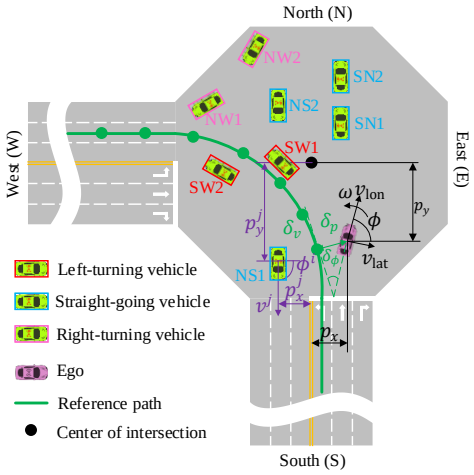


Fig. 3: State design in the scenario.

are designed as $Q = diag(0.04, 0.04, 0.01, 0.01, 0.1, 0.02)$, $R = diag(0.1, 0.005)$. The predictive horizon T is set to be 25, which is 2.5s in practical.

3) Constraint construction: 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 circles as illustrated by Fig. 4, where r_{veh} and r_{ego} are radii of circles of a vehicle and the ego. 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 rather than between only their CGs. Similar are the constraints between the ego and the road edge. Parameters for constraints: $M = diag(1, 1, 0, 0, 0, 0)$, $D_{\text{veh}}^{\text{safe}} = r_{\text{veh}} + r_{\text{ego}}$, $D_{\text{road}}^{\text{safe}} = r_{\text{ego}}$, where $r_{\text{ego}} = r_{\text{veh}} = 2.5$m. For the traffic light constraint, we convert it to constraints between ego and vehicles by placing two virtual vehicles on the stop line, as shown in Fig. 4.

4) Vehicle dynamics and prediction model: F_{ego} has been shown in (3), where all the vehicles parameters are displayed in Table I. Moreover, according to the type and position of the vehicle $j, j \in I$, the turning rate ω_{pred}^j in the prediction model is determined, as shown in Table II.

5) Training settings: We implement the offline training Algorithm 2 in an asynchronous learning architecture proposed in [34]. For value function and policy, we use a multi-layer perceptron (MLP) with 2 hidden layers, consisting of

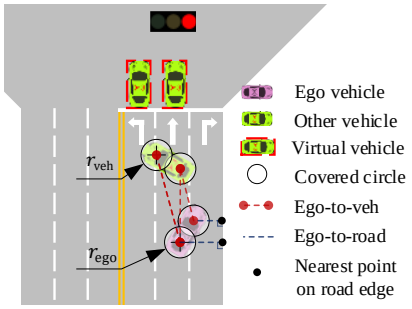


Fig. 4: Design of the state constraints.

TABLE I: Parameters for F_{ego}

Parameter	Meaning	Value
k_f	Front wheel cornering stiffness	-155495 [N/rad]
k_r	Rear wheel cornering stiffness	-155495 [N/rad]
L_f	Distance from CG to front axle	1.19 [m]
L_r	Distance from CG to rear axle	1.46 [m]
m	Mass	1520 [kg]
I_z	Polar moment of inertia at CG	2640 [kg·m^2]
Δt	System frequency	0.1 [s]

256 units per layer, with Exponential Linear Units (ELU) each layer [35]. The Adam method [36] with a polynomial decay learning rate is used to update all the parameters. Specific hyperparameter settings are listed in Table III. We train 5 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

Followed the Algorithm 1, the planned paths are shown in Fig. 5a. We also demonstrate the tracking and safety performances of different tasks during the training process in Fig. 5, indicated 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. Specially, 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.

In addition, we apply the trained policy of the left-turning task in the environment to visualize one typical case where a dense traffic and different phase of traffic light are included. Note that different from the training, we carry out all the

TABLE II: Parameters for F_{pred}

ω_{pred}^j		Position reletive to the intersection	
		Out of	Within
Vehicle type	Left-turning	0	$v_{\text{lon}}^j/26.875$
	Straight-going	0	0
	Right-turning	0	$-v_{\text{lon}}^j/15.625$

TABLE III: 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

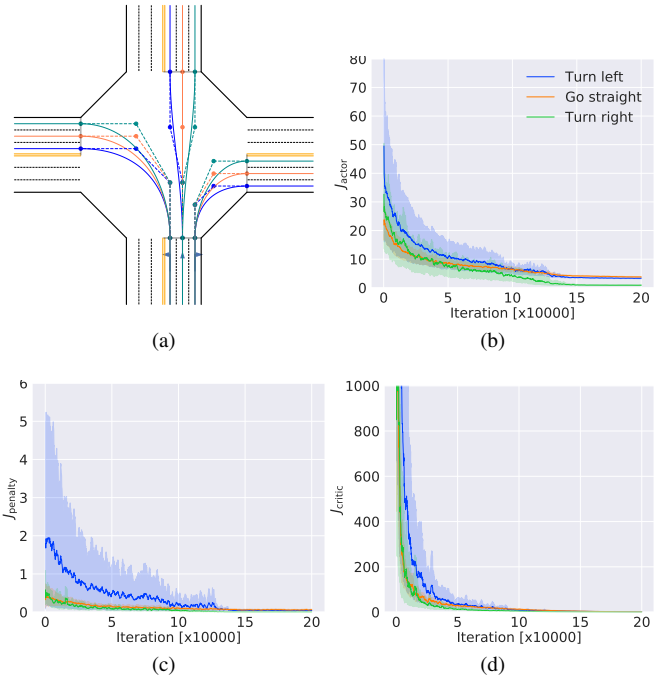


Fig. 5: Results of static path planning and dynamic optimal tracking. (a) Planned paths for each task. (b) Tracking performance during training process. (c) Safety performance during training process. (d) Value loss during training process. For (b)-(d), The solid lines correspond to the mean and the shaded regions correspond to 95% confidence interval over 5 runs.

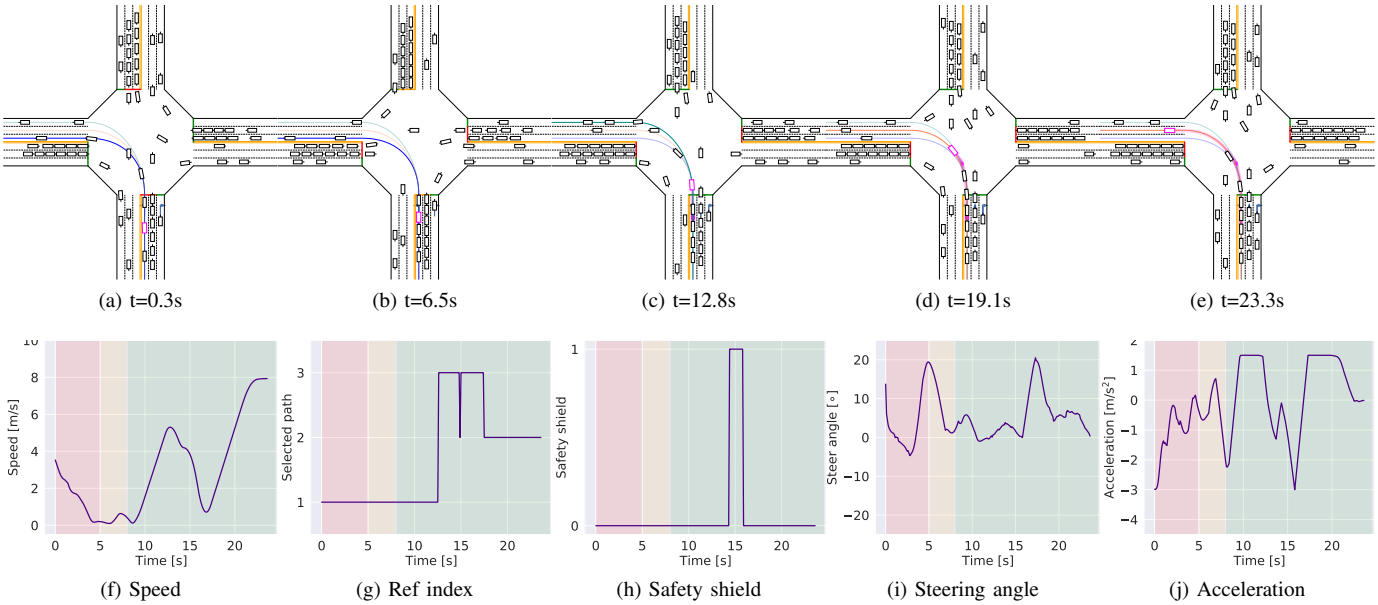
simulation tests on a 2.90GHz Intel Core i9-8950HK CPU. As shown in Fig. 6, when the traffic light is red, the ego pulls up to avoid collision and to obey the rule. Then when the light turns green, the ego starts itself off to enter the junction, where it meets several straight-going vehicles from the opposite direction. Therefore, the ego chooses the upper path and slows down to try 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-turning vehicles so that the velocity tracking error can be largely reduced. The computing time in each step is under 10ms, making our method considerably fast to be applied in the real world.

D. Experiment 1: Performance of GEP

To verify the control precision and computing efficiency of our model-based solver on constrained optimal problem, we conduct comparison with the classic Model Predictive Control (MPC), which utilizes the receding horizon optimization online and can deal with constraints explicitly. Formally, the problem (1) is defined on one certain path, and thus MPC method solves the same number of problems as that of candidate paths and calculate cost function of each path respectively. Then optimal path is selected by the minimum cost function and its corresponding action is used as the input signal of ego vehicle. Here we adopt the Ipopt solver to obtain the exact solution of the constrained optimal control problem, which is an open and popular package to solve nonlinear optimization problem. Fig. 7 demonstrates the comparison of our algorithm on control effect and computation time. Results show that the optimal path of 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. However, there exists the obvious difference in computation time that our method can output the actions within 10ms while MPC will take 1000ms to perform that. Although MPC can find the optimal solution by its 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

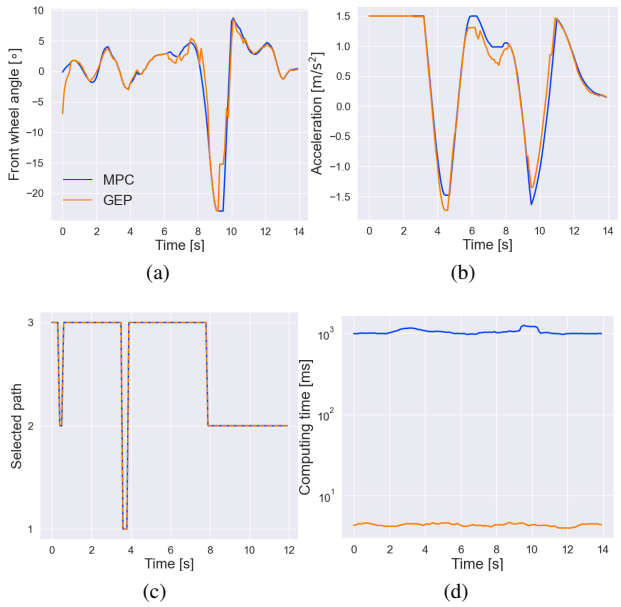
We compare our method with a rule-based method which adopts A* algorithm to generate a feasible trajectory and a PID controller to track it [19], as well as a model-free RL method which uses a punish and reward system to learn a policy for maximizing the long-term reward (If the ego vehicle passes the intersection safely, a large positive reward 100 is given, otherwise -100 is given wherever a collision happens) [37]. We choose six indicators including computing time, comfort, travel efficiency, collisions, failure rate and driving compliance to evaluate the three algorithms. Comfort is reflected by the mean root square of lateral and longitudinal acceleration, i.e., $I_{\text{comfort}} = 1.4\sqrt{(a_x^2) + (a_y^2)}$. 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 1 seconds and driving compliance shows times of breaking red light. The results of 100 times simulation are shown in Table IV. Rule-based method is more likely to stop and wait for other vehicles, leading to higher passing time but better safety performance. However, it tends to take 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



(a) t=0.3s (b) t=6.5s (c) t=12.8s (d) t=19.1s (e) t=23.3s

(f) Speed (g) Ref index (h) Safety shield (i) Steering angle (j) Acceleration

Fig. 6: Visualization of one typical episode driven by the trained policy.



(a) (b) (c) (d)

Fig. 7: Comparison with MPC. (a) Front wheel angle. (b) Acceleration. (c) Optimal path. (d) Computing time

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 performance by showing a group of complex and intelligent driving trajectories (Fig. 8), which demonstrates the potential

TABLE IV: 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

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 located at (31°08′13″N, 120°35′06″E), shown in Fig. 9a. The east-west street has a eight-lane dual carriageway from both directions, while the north-south street has a only four-lane dual carriageway. The detailed size and functionality of each lane are illustrated in Fig. 9b. To comply with legal requirements, we did not utilize the traffic flow and traffic signals of the real intersection. Instead, these traffic elements are designed and provided by SUMO. The experiment vehicle is CHANA CS55 equipped with RTK GPS, which realizes precise localization of the ego vehicle. In each time step, the ego states gathered from the CAN bus and the RTK are mapped into SUMO traffic to obtain the current traffic states including the states of surrounding vehicles and traffic signals. Then they both are sent to the industrial computer, where the trained policy and value functions are embedded. The computer is KMDA-3211 with a 2.6 GHz Intel Core I5-6200U CPU. Processed by our online algorithm, the safe actions

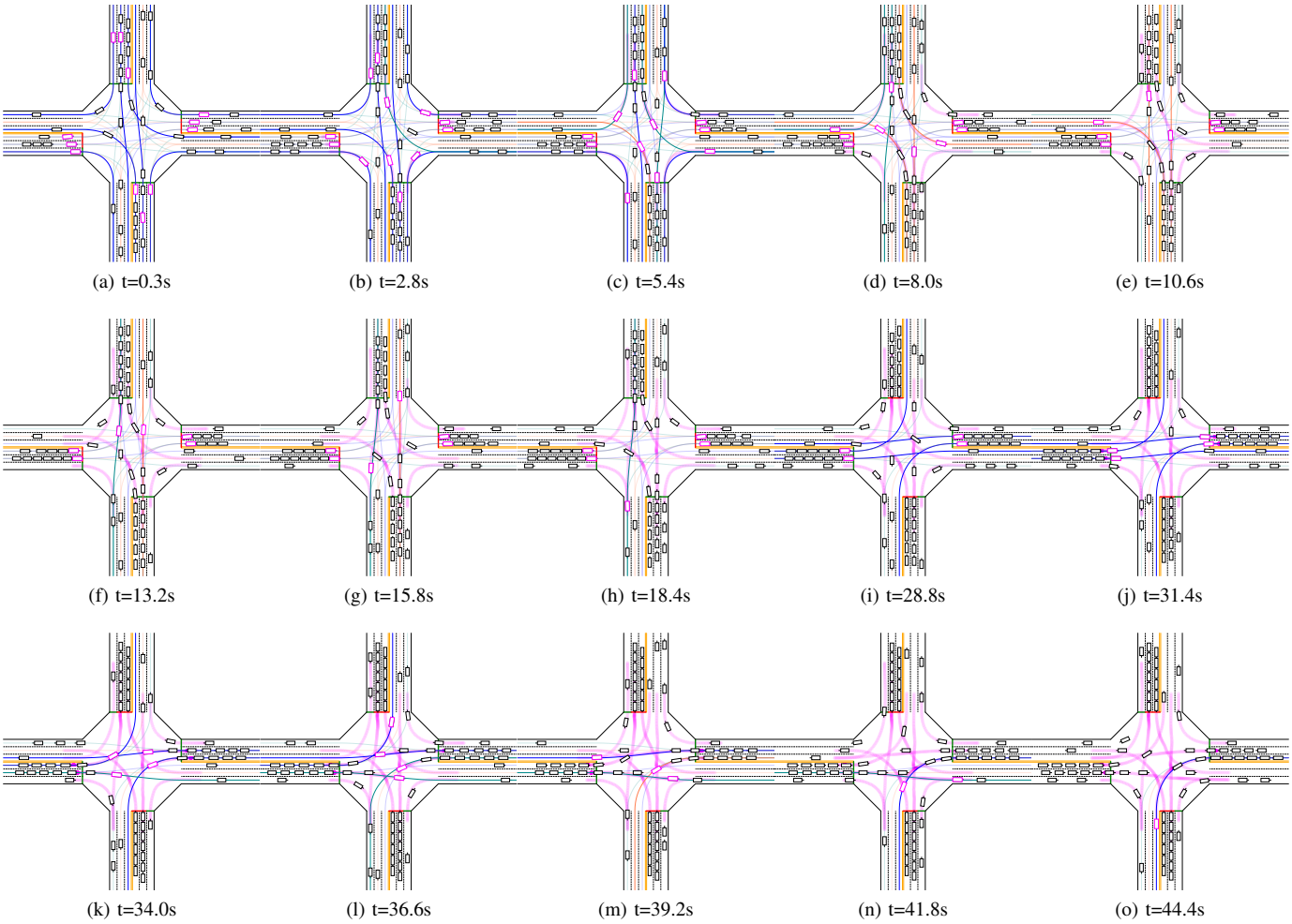
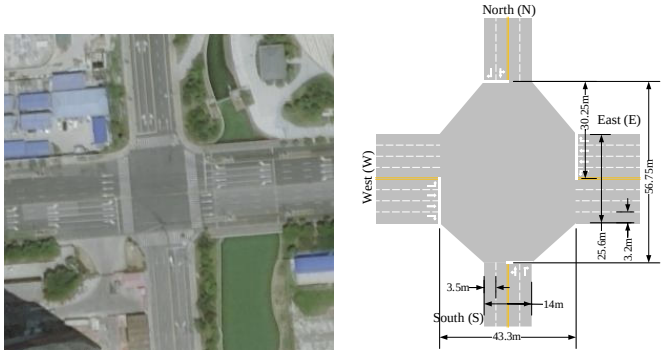


Fig. 8: Demonstration of the distributed control carried out by the trained policies.

including the steering angle and the expected acceleration are then delivered from the CAN bus to the real vehicle for its real time control. The experiment settings are illustrated in Fig. 10. Similar to section V, the ego vehicle enters the intersection from south, and is required to complete the same tasks (turn left, go straight, and turn right) under the signal control and a dense traffic flow.



(a) (b)

Fig. 9: The intersection for the real test.

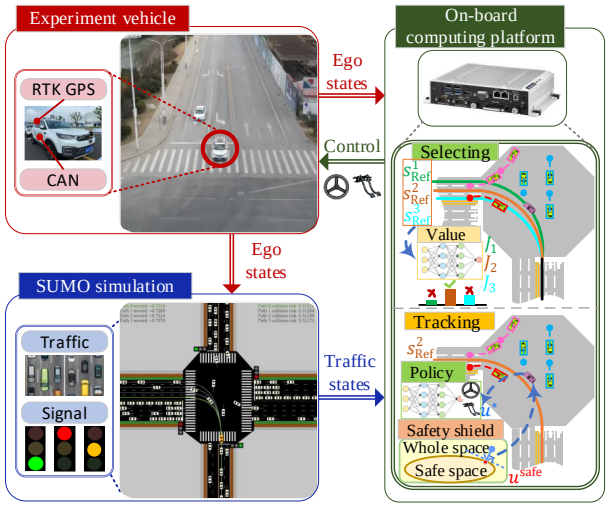
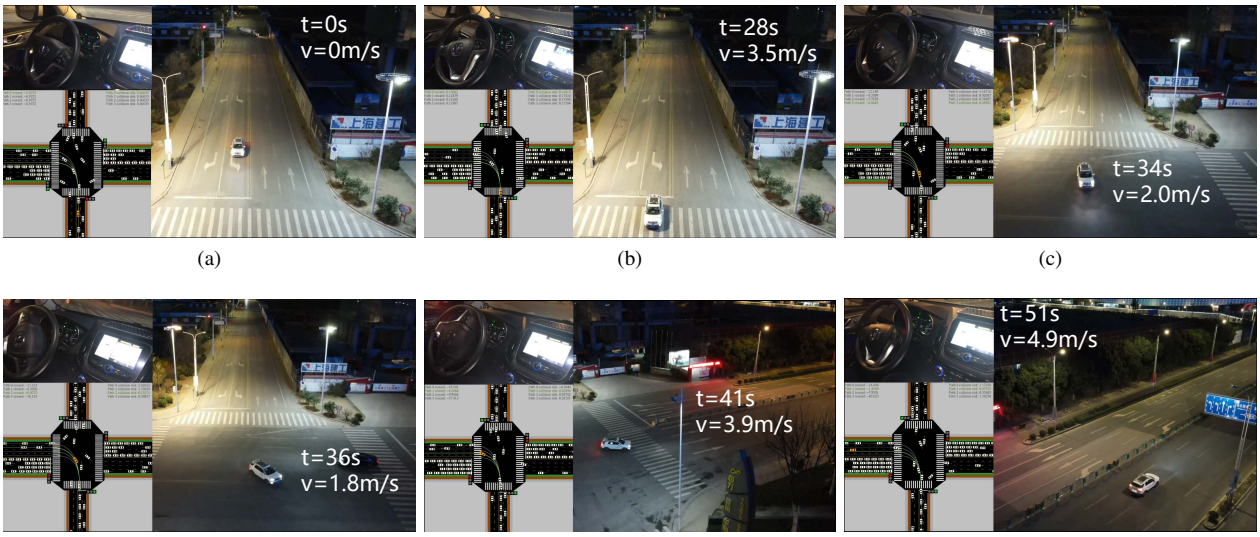


Fig. 10: Diagram of the real-world road test.

B. *Experiment 1: Functionality verification*

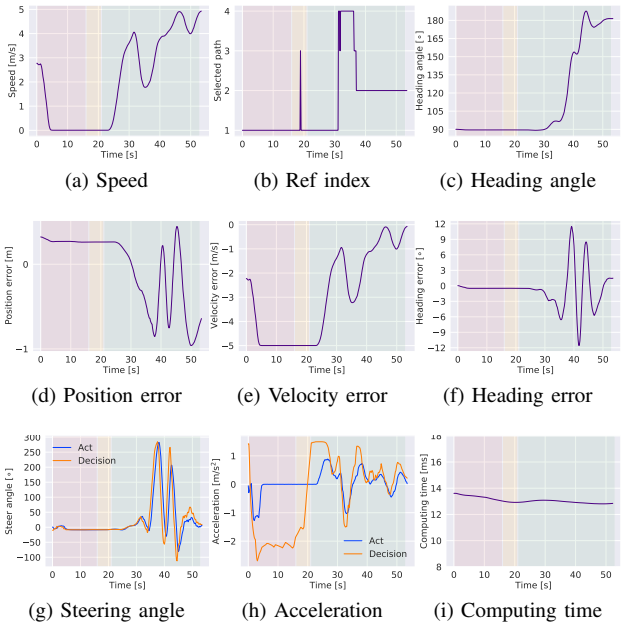
This experiment aims to verify the functionality of the IDC framework under different tasks and scenarios. In total, nine



(a) (b) (c)

(d) (e) (f)

Fig. 11: Featured time steps of 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

Fig. 12: Key parameters in the left run.

runs were carried out, three for each task. In each run, the ego vehicle is initialized before the south entrance with random states, meanwhile, the surrounding vehicles and signals are also initialized randomly. Following the diagram in Fig. 10, the run keeps going on until the ego passes the intersection successfully, i.e., 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).

Since the left-turning is the most complex task with the most potential interactions with surrounding vehicles, we visualize one of the left runs by snapshotting its featured time steps shown in Fig. 11 and drawing the key parameters in Fig. 12. At the beginning, the ego pulls up before the stop line, waiting for the green light (Fig. 11a). When that comes, the ego accelerates into the intersection to reduce the velocity tracking error (Fig. 11b). In the center of the intersection, it encounters a straight-going vehicle with high speed from the opposite direction. In order to avoid collision, the ego slows itself down and switches to the path 4, with which it is able to bypass the vehicle from back (Fig. 11c). However, another straight-going vehicle comes over after the previous one passes through, but with a relative 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 the path 4 to the path 3 and finally the path 2, to minimize the tracking errors (Fig. 11c and Fig. 11d). Following the path 2, the ego finally passes the intersection successfully. The computing time of all step is within 15ms, showing the superior of our method in terms of the online computing efficiency.

C. Experiment 2: Robustness to noise

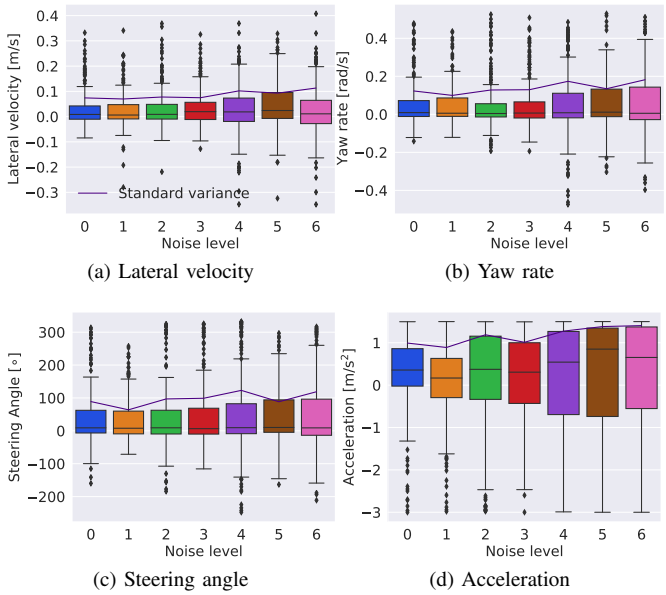
This experiment aims to compare the driving performance under different levels of noises added manually to verify the robustness of the trained policies. Referring to [38], we take similar measure to divide the noises into 7 levels, i.e. 0-6, where all the noises are in form of Gaussian white noise with different variances varying with the level and are applied in several dimensions of RL states, as shown in Table V.

We choose the left-turning task to perform seven experiments, one for each noise level in the Table V, to show its influence on the effect of the proposed framework. 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. 13. Our method is rather robust to low level noises (0-3) in which the distribution

TABLE V: Noise level 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

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 large variance of the steering angle, which further leads to the swing of the vehicle body. But nevertheless the stability bounds always remain in a reasonable range, proving the robustness of the proposed method.



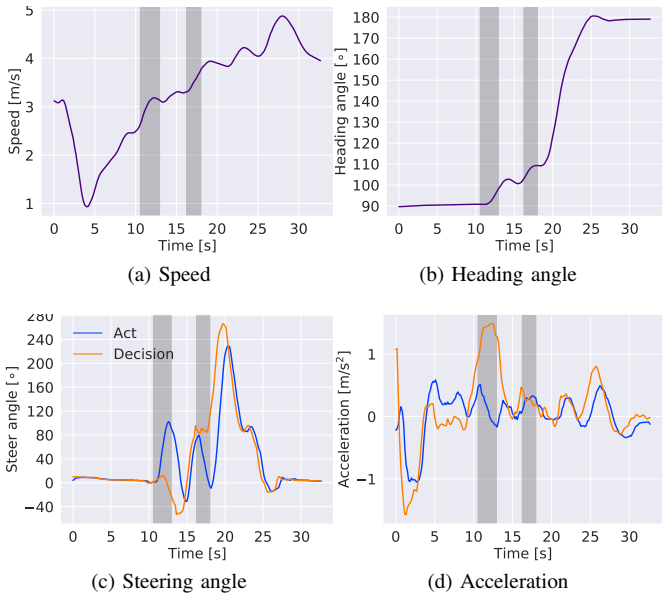
(a) Lateral velocity (b) Yaw rate

(c) Steering angle (d) Acceleration

Fig. 13: States and actions in different noise level.

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-turning case, in which we perform two times of human intervention on the steer wheel, and then switch to the autonomous driving mode. We draw the states of the ego vehicle in Fig. 14, where the colored region is when the human disturbance is acted on. The first one is acted on 10s, 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 turn the steering wheel right immediately to correct the excessive ego heading. The second one happens at 16s, 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 take-over, 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.



(a) Speed (b) Heading angle

(c) Steering angle (d) Acceleration

Fig. 14: States and actions under human disturbance.

VII. Conclusion

In this paper, we propose integrated decision and control (IDC) framework for automated vehicles, for the purpose of building an interpretable learning system with high online computing efficiency and applicability among different driving tasks and scenarios. The framework decomposes the driving task into the static path planning and the dynamic optimal tracking hierarchically. The former is in charge of generating multiple paths only 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 take dynamic obstacles into consideration, and then solves it offline by a model-based RL algorithm we propose to seek an approximate solution of an OCP in form of neural networks. 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 due to online optimizations. We verify our framework in both simulation and in a real world intersection. Results show that our method has 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. In the future, we will place efforts on designing more general state represen-

tations to extend the tracking ability of the lower layer among paths in different tasks or even 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 Transactions on intelligent vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

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

[3] G. Li, S. E. Li, B. Cheng, and P. Green, "Estimation of driving style in naturalistic highway traffic using maneuver transition probabilities," *Transportation Research Part C: Emerging Technologies*, vol. 74, pp. 113–125, 2017.

[4] L. Sun, W. Zhan, and M. Tomizuka, "Probabilistic prediction of interactive driving behavior via hierarchical inverse reinforcement learning," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2111–2117.

[5] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 961–971.

[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 Transactions on Intelligent Transportation Systems*, 2019.

[7] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke *et al.*, "Junior: The stanford entry in the urban challenge," *Journal of field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[8] M. Olsson, "Behavior trees for decision-making in autonomous driving," 2016.

[9] A. Gray, Y. Gao, T. Lin, J. K. Hedrick, H. E. Tseng, and F. Borrelli, "Predictive control for agile semi-autonomous ground vehicles using motion primitives," in *2012 American Control Conference (ACC)*. IEEE, 2012, pp. 4239–4244.

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

[11] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.

[12] U. Lee, S. Yoon, H. Shim, P. Vasseur, and C. Demonceaux, "Local path planning in a complex environment for self-driving car," in *The 4th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent*. IEEE, 2014, pp. 445–450.

[13] 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, 2015.

[14] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[15] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *IEEE Transactions on control systems technology*, vol. 17, no. 5, pp. 1105–1118, 2009.

[16] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," *Robotics Science and Systems VI*, vol. 104, no. 2, 2010.

[17] J. Shah, M. Best, A. Benmimoun, and M. L. Ayat, "Autonomous rear-end collision avoidance using an electric power steering system," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 229, no. 12, pp. 1638–1655, 2015.

[18] H. Mouhagir, V. Cherfaoui, R. Talj, F. Aioun, and F. Guillemard, "Trajectory planning for autonomous vehicle in uncertain environment using evidential grid," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 12 545–12 550, 2017.

[19] L. Xin, Y. Kong, S. E. Li, J. Chen, Y. Guan, M. Tomizuka, and B. Cheng, "Enable faster and smoother spatio-temporal trajectory planning for autonomous vehicles in constrained dynamic environment," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 235, no. 4, pp. 1101–1112, 2021.

[20] S. Eben Li, K. Li, and J. Wang, "Economy-oriented vehicle adaptive cruise control with coordinating multiple objectives function," *Vehicle System Dynamics*, vol. 51, no. 1, pp. 1–17, 2013.

[21] S. Li, K. Li, R. Rajamani, and J. Wang, "Model predictive multi-objective vehicular adaptive cruise control," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 3, pp. 556–566, 2010.

[22] S. E. Li, "Reinforcement learning and control," 2020, tsinghua University: Lecture Notes. http://www.idlab-tsinghua.com/thulab/labweb/publications.html.

[23] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–18, 2021.

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

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

[26] D. C. K. Ngai and N. H. C. Yung, "A multiple-goal reinforcement learning method for complex vehicle overtaking maneuvers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 509–522, 2011.

[27] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[28] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016*, 2016.

[29] 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 Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 12 597–12 608, 2020.

[30] J. Chen, B. Yuan, and M. Tomizuka, "Model-free deep reinforcement learning for urban autonomous driving," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 2765–2771.

[31] Q. Ge, S. E. Li, Q. Sun, and S. Zheng, "Numerically stable dynamic bicycle model for discrete-time control," *arXiv preprint arXiv:2011.09612*, 2020.

[32] Z. Allen-Zhu, Y. Li, and Z. Song, "A convergence theory for deep learning via over-parameterization," in *International Conference on Machine Learning*. PMLR, 2019, pp. 242–252.

[33] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. [Online]. Available: https://elib.dlr.de/124092/

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

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

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

[37] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1861–1870.

[38] J. Duan, "Study on distributional reinforcement learning for decision-making in autonoumou driving," Ph.D. dissertation, Tsinghua University, Beijing, China, 2021.