Policy Iteration Based Approximate Dynamic Programming Toward Autonomous Driving in Constrained Dynamic Environment

Ziyu Lin, Jun Ma, Jingliang Duan, *Member, IEEE*, Shengbo Eben Li, *Senior Member, IEEE*, Haitong Ma, Bo Cheng, and Tong Heng Lee

***Abstract*— In the area of autonomous driving, it typically brings great difficulty in solving the motion planning problem since the vehicle model is nonlinear and the driving scenarios are complex. Particularly, most of the existing methods cannot be generalized to dynamically changing scenarios with varying surrounding vehicles. To address this problem, this development here investigates the framework of integrated decision and control. As part of the modules, static path planning determines the reference candidates ahead, and then the optimal path-tracking controller realizes the specific autonomous driving task. An innovative and effective constrained finite-horizon approximate dynamic programming (ADP) algorithm is herein presented to generate the desired control policy for effective path tracking. With the generalized policy neural network that maps from the state to the control input, the proposed algorithm preserves the high effectiveness for the motion planning problem towards changing driving environments with varying surrounding vehicles. Moreover, the algorithm attains the noteworthy advantage of alleviating the typically heavy computational loads with the mode of offline training and online execution. As a result of the utilization of multi-layer neural networks in conjunction with the actor-critic framework, the constrained ADP method is capable of handling complex and multidimensional scenarios. Finally, various simulations have been carried out to show that the constrained ADP algorithm is effective.**

***Index Terms*— Autonomous driving, approximate dynamic programming, motion planning, constrained optimization, reinforcement learning.**

Manuscript received 27 September 2021; revised 30 April 2022 and 28 December 2022; accepted 12 January 2023. Date of publication 26 January 2023; date of current version 8 May 2023. This work was supported in part by the National Key R&D Program of China under Grant 2022YFB2502901; in part by the Tsinghua University-Toyota Joint Research Center for AI Technology of Automated Vehicle; and in part by NSF China under Grant U20A20334, Grant 52072213, and Grant 52202487. The Associate Editor for this article was I. Papamichail. *(Corresponding author: Shengbo Eben Li.)*

Ziyu Lin, Shengbo Eben Li, Haitong Ma, and Bo Cheng are with the School of Vehicle and Mobility, Tsinghua University, Beijing 100084, China (e-mail: linzy17@mails.tsinghua.edu.cn; lishbo@tsinghua.edu.cn; maht19@mails.tsinghua.edu.cn; chengbo@tsinghua.edu.cn).

Jun Ma is with the Robotics and Autonomous Systems Thrust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China, and also with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong SAR, China (e-mail: jun.ma@ust.hk).

Jingliang Duan is with the School of Mechanical Engineering, University of Science and Technology Beijing, Beijing 100083, China (e-mail: duanjl@ustb.edu.cn).

Tong Heng Lee is with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117583 (e-mail: eleleeth@nus.edu.sg).

Digital Object Identifier 10.1109/TITS.2023.3237568

I. Introduction

THE development and advent of autonomous vehicles are promising to achieve a safer and more coordinated transportation network [1]. As an indispensable and crucial component in autonomous driving problems, motion planning has attracted substantial attention in recent researches. Basically, it aims to determine valid control inputs that enable the vehicle to follow along a feasible trajectory without collision. However, in most of the autonomous driving scenarios, obtaining the solution to the motion planning problem is challenging because of the nonlinearity of the vehicle model, complexity of the driving scenarios, and so on [2].

Generally, there are two major categories for motion planning methods, which consist of methods based on optimization and methods based on learning. Serving as one of the key underpinning basis of optimization-based motion planning methods, the methodology of Model Predictive Control (MPC) is widely adopted for autonomous driving tasks, where the optimal control input can be generated in a receding horizon fashion by predicting the future evolution of the systems [3]. For example, Borrelli et al. [4] proposed a novel MPC scheme for autonomous steering systems, which achieved the task of tracking along the desired path stably, making double-lane changes while satisfying the physical constraints. Erlien et al. [5] presented the MPC approach with the shared control framework to avoid obstacles effectively. Siampis et al. [6] applied the Primal-Dual Interior-Point method to solve the MPC problem with nonlinear models. Dixit et al. [7] planned the safe trajectories to perform an autonomous overtaking maneuver in highways, combining MPC and the potential field method. Hang et al. [8] also utilized the combination for motion planning, and a series of simulations were carried out for typical lane-changing tasks.

Despite the broad use and high effectiveness of the MPC, two challenging issues are commonly encountered. Firstly, due to the online receding horizon control framework, the MPC only finds the optimal control input at the time stamp for the current driving scenario; and when the vehicle rolls out to the next state, different environments with varying surrounding vehicles can be encountered, then the optimal control input needs to be determined again [9], [10]. In this sense, the MPC is not straightforward to be generalized to dynamically changing scenarios with varying surrounding vehicles. Taking

the left-turn task at the road intersection as an example, different positions of the ego vehicle and surrounding vehicles refer to different scenarios; and in this case, the MPC must be executed every time when different scenarios are encountered because of the restriction of the MPC. Secondly, for more complex driving scenarios, the nonlinear vehicle model and the non-convexity constraints from collision-avoidance bring heavy burdens to the computation efficiency [11], [12], which has been a long-standing challenge in the existing literature. Particularly, in the tense traffic flow when more surrounding vehicles are involved, the computational burden in solving such a motion planning problem certainly becomes a matter of serious concern.

Alternatively, recent advances in artificial intelligence have provided the possibility of solving the motion planning problem by the learning-based method. Learning-based methods aim to learn the system model [13] or feasible control policies from the dynamically varying system states from the environment [14], [15], [16]. For example, Duan et al. [16] realized the motion control under highways with hierarchical reinforcement learning (RL), which achieved driving maneuvers by designing complicated reward functions instead of relying on the expert data. Makantasis et al. [17] utilized DDQN to solve the driving policy in high-way driving environments with multiple traffic participants in Sumo simulator. To enhance the sample efficiency, one of the model-based RL methods, Approximate Dynamic Programming (ADP) is brought to attention by the current research community [18]. It is established on the prior-known environmental dynamic model in the whole state-action space [19], [20]. ADP is efficient because the model provides its partial derivative information to facilitate gradient calculation. It reduces the occurrence of high-variance gradients coming from imperfect measurement and environment randomness.

Evidently thus, the ADP algorithm can be employed to overcome the two aforementioned challenges of the optimization-based method. On the one hand, the learned control policy obtained from the ADP algorithm is more suitable for the dynamically varying driving scenarios [21]. The ADP relies on the determination of an approximated control policy offline, and then the approximated control policy can be used in an online framework [21]. During the offline training process, different dynamic environments can be considered and the ADP aims to search for the optimal control policy covering the whole state space instead of merely the current state point, and thus it is expected to improve the generalization of the control policy appropriately [22], [23]. On the other hand, the ADP is expected to realize high computation efficiency attributing to the mode of offline training and online execution utilizing a low-dimensional parameterized function [24]. In this case, the heavy computational effort in online optimization to solve the optimal control problem can be relieved with offline training of the value and policy neural networks utilizing the ADP method. Furthermore, the proposition of the ADP addresses the curse of dimensionality of dynamic programming, and the learned policy is obtained by utilizing the proper approximation of the value function, such as neural networks [25]. Hence, the ADP method is more suitable for medium-sized or even large-scale scenarios, which are commonly encountered in the domain of autonomous driving when more surrounding vehicles are involved. It is notable that a series of seminal works have explored the use of the ADP method in autonomous driving. For instance, Zhao et al. [26] applied the ADP algorithm on autonomous vehicles to achieve good performance for the path following control problem. Moreover, Zhao et al. [27] utilized the ADP algorithm to seek a near-optimal control policy for the autonomous vehicle.

However, most ADP researches deal with rather simple driving tasks such as line-keeping or reference-tracking control on highways, where collision avoidance constraints are not considered. In fact, these constraints are inevitable in the motion planning tasks of autonomous vehicles in view of the highly dynamic urban road traffic. To the authors' best knowledge, in the sense of on-road autonomous driving motion planning problems, there is still a lack of very highly effective methods that can address the general form of constraints using the ADP. In fact, the introduced collision-avoidance constraints render great difficulty in the solving process of the ADP, because of the intractability in solving the constrained Hamilton-Jacobi-Bellman (HJB) equation with the implementation of policy iteration framework [28]. Lin et al. [29] rebuilt the finite-horizon ADP algorithm using the PI framework with reshaped HJB equation. However, it cannot deal with the system dynamic constraint and collision-avoidance constraint. Hence, it still remains an open problem to solve the constrained ADP problem in autonomous driving efficiently and effectively.

The work in this paper develops and presents an innovative methodology of a highly effective generic guideline to solve the constrained ADP problem as part of the motion planning task in autonomous driving, where a constrained ADP algorithm for the continuous-time finite-horizon systems is proposed with high efficiency and effectiveness. By leveraging the actor-critic framework, value and policy networks established the mapping between system states and value function and control policy. A series of simulations are conducted to achieve a vehicle-tracking task for the validation of the proposed ADP methodology. The main contributions of this paper are thus as follows:

1) The constrained ADP algorithm for a continuous-time finite-horizon system is proposed to produce the desired control input for dynamically varying driving scenarios. With the generalized policy, the proposed approach can be catered to changing driving environments with varying surrounding vehicle constraints.
2) The efficiency of the proposed motion planning algorithm based on the constrained ADP is greatly improved compared with the optimization-based method. The heavy computational burden in online optimizations is relieved by solving the constrained optimal control problem offline and then applying the value network and policy networks online to select and track the path.
3) The proposed finite-horizon constrained ADP algorithm is suitable for driving scenarios with high dimensionality and complexity, which attributes to the utilization of

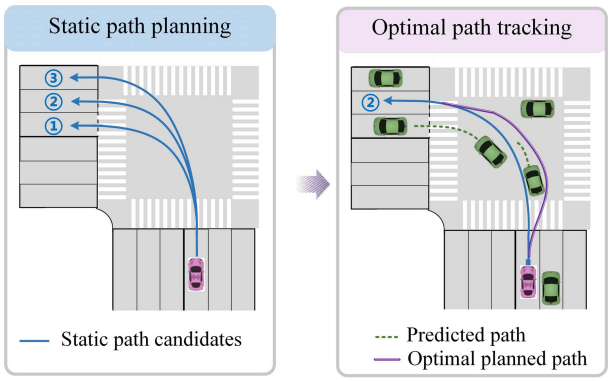

Fig. 1. The illustration of the integrated decision and control for motion planning.

multi-layer neural networks parameterizing both the approximated value and control policy upon the actor-critic framework.

The following shows the organization of the rest of the study. Section II presents the motion planning framework. Section III discusses the design of the static reference path candidates utilizing the cubic Bezier curve. Section IV formulates the path-tracking control problem. Section V presents the constrained ADP algorithm to solve the path-tracking control problem. Section VI shows the simulation results. The conclusion of this paper is given in Section VII.

II. Integrated Decision and Control Framework for Motion Planning

For the motion planning task, integrated decision-making and control is implemented, which achieves two functions separately: planning static paths and tracking the optimal path [30].

As shown in Fig. 1, the first module aims to pre-generate multiple path candidates considering static road structure information. For the sake of computing efficiency, we take no account of the dynamically varying information of surrounding vehicles in this module. The expected positions in the static paths are designed, and the desired velocity is given as constant. In this case, the advantage of this static path planning lies in the simplification and efficiency, so it is suitable to be applied in general driving scenarios with known parameters of the road.

The second module, i.e., optimal path tracking, is of great importance in this framework. It determines the optimal static path to follow, which is chosen from the pre-generated path candidates. We choose the path that minimizes the value function, which evaluates the tracking performance. Meanwhile, the path can be dynamically adjusted during the tracking process. Moreover, this module also decides the optimal control to achieve the tracking task. In essence, the path-tracking task can be formulated as a constrained optimal control problem. The operation of the steering and acceleration is chosen as the control input, which is solved by minimizing value functions related to the tracking errors. In this paper, the constrained ADP algorithm is adopted to determine the optimal policy based on the actor-critic framework. Moreover, two neural networks are adopted to improve computational efficiency. In conclusion, the design of the integrated decision and control is suitable for real implementation with high-efficiency demand.

III. Static Path Planning

In this work, each task is assigned three paths according to the lane number of exits. The candidates of static paths are generated by the cubic Bezier curve, which needs four known points about the road map. Each reference can be determined by the vector $(R_x, R_y, R_\varphi, R_v)$ utilizing

$$\begin{aligned} R_x(\gamma) &= R_{x_0}(1-\gamma)^3 + 3R_{x_1}\gamma(1-\gamma)^2 \\ &\quad + 3R_{x_2}\gamma^2(1-\gamma) + R_{x_3}\gamma^3 \\ R_y(\gamma) &= R_{y_0}(1-\gamma)^3 + 3R_{y_1}\gamma(1-\gamma)^2 \\ &\quad + 3R_{y_2}\gamma^2(1-\gamma) + R_{y_3}\gamma^3 \\ R_\varphi(\gamma) &= \arctan\left(\frac{R_y(\gamma+1) - R_y(\gamma)}{R_x(\gamma+1) - R_x(\gamma)}\right) \\ R_v &= v_{\text{exp}}, \end{aligned} \tag{1}$$

where R_x, R_y, R_φ, R_v represent the position in the X-coordinate, the position in the Y-coordinate, the heading angle, and the velocity of the reference point on the static paths, respectively. The parameter $\gamma \in [0, 1]$ influences the shape of the Bezier curve. Four solid feature points (R_{x_0}, R_{y_0}), (R_{x_1}, R_{y_1}), (R_{x_2}, R_{y_2}), and (R_{x_3}, R_{y_3}) represent the starting point, two intersection points, and the destination point, respectively, which are shown in Fig. 2. Two intersection points are chosen on the extended distance d_1 and d_2 from the other two points, which are manually set considering the size of the intersection and the consistency of curvature. Meanwhile, we design an expected velocity v_{exp} for each path candidate, which is a fixed value for the whole trajectory.

Simplified path candidates are generated without considering collision avoidance, which is efficient. Moreover, the planned paths only provide references for the path tracking module, and the actual trajectory may be different from the static path due to further considerations on dynamic obstacles.

IV. Path-Tracking Control Problem Formulation

A. Dynamic Model of Ego Vehicle

The dynamic model is important to formulate the path-tracking control problem. To express the vehicle model explicitly, the vector of the states is defined as:

$$x_{ego} = \left[v_x, v_y, r, p_x, p_y, \varphi\right]^\top, \tag{2}$$

where v_x, v_y represent the longitudinal speed and lateral speed; r denotes the yaw rate; p_x, p_y is the position of the vehicle in the X-coordinate and Y-coordinate; φ is the heading angle. Meanwhile, the vector for the input u consist of δ, and a_{acc}, where δ represents the steering angle and a_{acc} denotes the acceleration.

To describe the vehicle dynamic model, the two-degree-of-freedom bicycle model is employed in this study. Hence,

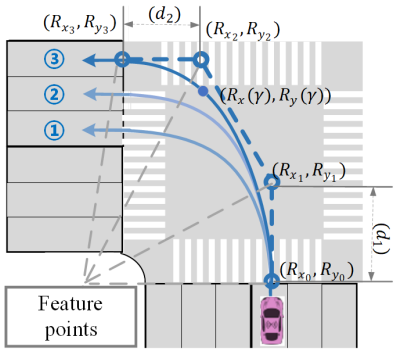

Fig. 2. Static path candidates generated by cubic Bezier curve.

the ego vehicle's dynamic model in continuous time can be derived in reference to a geodetic coordinate

$$\dot{x}_{ego} = f_{ego}\left(x_{ego}, u\right) = \begin{bmatrix} a_{\text{acc}} \\ \left(\frac{ak_1 - bk_2}{mv_x} - v_x\right) r + \frac{k_1 + k_2}{mv_x} v_y - \frac{k_1}{m}\delta \\ \frac{a^2k_1 + b^2k_2}{I_{zz}v_x} r + \frac{ak_1 - bk_2}{I_{zz}v_x} v_y - \frac{ak_1}{I_{zz}}\delta \\ v_x \cos\varphi - v_y \sin\varphi \\ v_x \sin\varphi + v_y \cos\varphi \\ r \end{bmatrix}, \quad (3)$$

where a, b denote the distance between the mass center to the front and rear axle; k_1 are the cornering stiffness of the front wheel; k_2 is the stiffness of the rear wheel; m denotes the mass; I_{zz} represents the polar moment of inertia. In this study, the linear tire sideslip force model is adopted. To guarantee the stability of the vehicle stability, the yaw rate r and the slip angle α are limited in the boundary, i.e., $-\frac{\mu_r g}{v_x} \le r \le \frac{\mu_r g}{v_x}$, $-\frac{3\mu_\# F_{z\#}}{C_\#} \le \alpha_\# \le \frac{3\mu_\# F_{z\#}}{C_\#}$, where # represents the front and the rear wheel; $\mu_\#$ denotes the lateral friction coefficient ($\mu_\# = \frac{\sqrt{(\mu F_{z\#})^2 - (F_{x\#})^2}}{F_{z\#}}$); F_x represents the tire forces in the longitudinal direction. Loads of wheels are required, which can be calculated as $F_{zf} = \frac{b}{a+b}mg$, $F_{zr} = \frac{a}{a+b}mg$, $\left[F_{xf}, F_{xr}\right] = \begin{cases} [0, m\ a_{\text{acc}}], & a_{\text{acc}} \ge 0 \\ \left[\frac{ma_{\text{acc}}}{2}, \frac{ma_{\text{acc}}}{2}\right], & a_{\text{acc}} < 0. \end{cases}$

In view of this optimal control problem, the tracking errors concerning the known reference trajectory are considered. Therefore, three new state variables of the system are introduced

$$x_{error} = [\Delta p_d, \Delta\varphi, \Delta v_x]^\top. \quad (4)$$

Here, Δp_d represents the position error, $\Delta\varphi$ denotes the heading angle error, and Δv_x means the velocity error. They are defined as

$$\begin{aligned} \Delta p_d &= \sqrt{(p_x - R_x)^2 + (p_y - R_y)^2}, \\ \Delta\varphi &= \varphi - R_\varphi, \\ \Delta v_x &= v_x - R_v. \end{aligned}$$

With the assumption that $\dot{\phi}_r = 0$, the following error dynamic model is obtained

$$\dot{x}_{error} = f_{error}\left(x_{error}, u\right) = \begin{bmatrix} v_x \sin\varphi + v_y \cos\varphi \\ r \\ a_{\text{acc}} \end{bmatrix}. \quad (5)$$

B. Dynamic Model of Surrounding Vehicles

Apart from the ego vehicle, the traffic participants in the autonomous driving problem also include the surrounding vehicles. Here, the state vector for the surrounding vehicles is denoted as

$$x_j = \left[p_{x_j}, p_{y_j}, v_j, \varphi_j\right]^\top, \quad (6)$$

where p_{x_j}, p_{y_j} represent the position of the j-th surrounding vehicle center in X-coordinate and Y-coordinate, respectively, $j \in [0, N_{veh}]$, N_{veh} is the number of the surrounding vehicles. v_j, φ_j denote the speed and the heading angle of the j-th surrounding vehicle, respectively.

The trajectory of each surrounding vehicle is designed to go straight, turn left, or turn right. A kinematics model is adopted, where the predicted position states are determined by the velocity, and the heading angle is dependent on the yaw rate, i.e.,

$$\dot{x}_j = f_j\left(x_j, u\right) = \begin{bmatrix} v_j \sin\varphi_j \\ v_j \cos\varphi_j \\ 0 \\ r_j \end{bmatrix}, \quad (7)$$

where r_j is chosen depending on whether the surrounding vehicle goes straight or turns. When it goes straight, the heading angle keeps constant, i.e., $r_j = 0$. When it turns left or right, $r_j = \frac{v_j}{R^*}$, where R^* is the radius of the intersection.

C. Safety Constraints

To prevent collisions with the surrounding vehicles and the road boundary when tracking the desired reference trajectory, the safety constraints are defined as

$$\begin{aligned} (p_{x^\#} - p_{x_j^*})^2 + (p_{y^\#} - p_{y_j^*})^2 - d_{\text{safe}}^2 &\ge 0, \\ (p_{x^\#} - p_{x_{\text{road}}})^2 + (p_{y^\#} - p_{y_{\text{road}}})^2 - d_{r_{\text{safe}}}^2 &\ge 0, \end{aligned} \quad (8)$$

where $(p_{x^\#}, p_{y^\#})$ and $(p_{x_j^*}, p_{y_j^*})$ are the centers of the circles for ego and surrounding vehicles, respectively, the superscripts $\#, * \in \{f, r\}$ represent the safety circle in the front or rear. $(p_{x_{\text{road}}}, p_{y_{\text{road}}})$ denote the closest point to the boundary of the road. Each vehicle is regarded as two circles. Meanwhile, r_{ego} and r_{veh} represent the radius of the ego vehicle and the surrounding vehicle, respectively. In addition, d_{safe} represents the safe distance between vehicles, and $d_{r_{\text{safe}}}$ represents the safe distance between the vehicle and the road boundary.

The number of constraints in (8) is dependent on the number of surrounding vehicles. For brevity, the safety state constraints can be expressed in the general form as

$$h(x, u) \ge 0. \quad (9)$$

D. Problem Formulation

Generally, the motion planning problem for the autonomous vehicle can be transferred into a continuous-time optimal control problem. As one important part, the corresponding finite-horizon value function is introduced as

$$V^{\pi}(x(t),t)=\int_{t}^{T} l(x(\tau),u(\tau))d\tau, \tag{10}$$

where $x=\left[x_{ego},x_{error},x_j\right]^{\top}\in\Omega\subset\mathbb{R}^q$ represents the state vector ($q=9+N_{veh}*4$); the dimension of the control input $u\in\mathbb{R}^m$ is set two ($m=2$); the current time t satisfies $t\in[0,T]$; T reflects the predicition horizon of the problem; the utility function l is defined nonnegative. Meanwhile, the policy π aims to map the state-time pair to the control input for $\tau\in[t,T]$, i.e.,

$$u(\tau)=\pi(x(\tau),\tau). \tag{11}$$

The utility function mainly considers the position-tracking error, velocity-tracking error, and control input. Notably, the input of the steering wheel is taken into account due to the effect on safety and driving comfort. As for the acceleration input, it influences fuel consumption and driving comfort. Therefore, the utility function l at each time τ in the predictive horizon is formulated

$$l(x,u)=q_1\left\|M_d x\right\|^2+q_2\left\|M_v x\right\|^2+r_1u^T M_\delta u+r_2u^T M_a u, \tag{12}$$

where $\|\cdot\|$ represents the Euclidean norm operator; q_1,q_2 represents the weighted parameters of error on position and velocity; the parameters r_1 and r_2 weight the input importance of the steering and acceleration; $M_d x=\Delta p_d$, $M_v x=\Delta v_x$, then we have $M_\delta=\begin{bmatrix}0&0\\0&1\end{bmatrix}$ and $M_a=\begin{bmatrix}1&0\\0&0\end{bmatrix}$.

The continuous-time-constrained optimal control problem aims to find a control policy that achieves the minimization of the value function for all x. Therefore, we formulate the optimal control problem as

$$\begin{aligned}V^*(x(t),t)=\min_{u(\tau)}V(x,t)=\min_{u}\int_{t}^{T}l(x(\tau),u(\tau))\mathrm{d}\tau\\ \text{s.t. }\ \dot{x}(\tau)=f(x(\tau),u(\tau))\\ h(x(\tau),u(\tau))\geq 0\end{aligned} \tag{13}$$

where $f=\left[f_{ego},f_{error},f_j\right]^{\top}$ is the controllable dynamic model.

To solve the solution to (13), Hamiltonian function is firstly introduced as $H\left(x,u,\frac{\partial V^{\pi}(x,t)}{\partial x^{\top}}\right)=l(x,\pi(x,t))+\frac{\partial V^{\pi}(x,t)}{\partial x^{\top}}f(x,\pi(x,t))$. Here, x and u are used as the abbreviations for $x(t)$ and $u(t)$. Referring to [19], the self-consistency condition for (13) is obtained by taking the differentiation of (10) $H\left(x,u,\frac{\partial V^{\pi}(x,t)}{\partial x^{\top}}\right)=-\frac{\partial V^{\pi}(x,t)}{\partial t}$. In this case, if a control policy π is known, the value can be solved correspondingly.

Considering the principle of dynamic programming, solving the input sequence can be transferred into solving one single control action. Referring to [29], we can obtain the partial derivate of the value with respect to time $\frac{\partial V^{\pi}(x,t)}{\partial t}=-l\left(x^{\pi}(T),u(T)\right)+\zeta^{\pi}$. It includes two terms on the right-hand side of the equation: the terminal-time utility function and ζ^{π}. ζ^{π} is related to $\mathrm{d}H\left(x,\pi(x,\tau),\frac{\partial V^{\pi}(x(\tau),\tau)}{\partial x(\tau)}\right)/\mathrm{d}u(\tau)$. Furthermore, if $\mathrm{d}H/\mathrm{d}u=0$ (when u is optimal), it follows

$$\frac{\partial V^{\pi}(x,t)}{\partial t}=-l(x^{\pi}(T),u(T)). \tag{14}$$

Thereafter, the optimal control action for (13) needs to meet the finite-horizon HJB equation with the constraints taken into account

$$\begin{aligned}\min_{\pi}\left\{l(x,u)+\frac{\partial V^*(x,t)}{\partial x^{\top}}f(x,u))\right\}=l(x^*(T),u^*(T))\\ -\zeta^{\pi^*}\\ \text{s.t. }\ \dot{x}(t)=f(x(t),u(t))\\ h(x(\tau),u(\tau))\geq 0\end{aligned} \tag{15}$$

V. Constrained ADP for Path-Tracking Control

A. Constrained Finite-Horizon Policy Iteration

The essence of solving the optimal tracking control problem is to handle the HJB equation (15). Our proposed method utilizes the policy iteration technique to iteratively approximate value and control policy. Two alternating stages are included in this framework: (1) Unconstrained policy evaluation (PEV) and (2) Constrained policy improvement (PIM). In terms of the unconstrained PEV, it seeks to find the value concerning the current policy by calculating the following self-consistency condition

$$l(x,u^k)+\frac{\partial V^{\pi^k}}{\partial x^{\top}}f(x,u^k)=l(x^{\pi^k}(T),u^k(T)-\zeta^{\pi^k} \tag{16}$$

If the current policy π^k is feasible in a certain region, one can find its corresponding value. Hence, the PEV stage doesn't consider the constraints.

In the stage of the constrained PIM, a better policy is solved with the known value $V^{\pi^k}(x,t)$. The minimization of the Hamiltonian is a variant of greedy search, which provides the updating rule for the policy

$$\begin{aligned}\pi^{k+1}(x,t)=\arg\min_{\pi}H\left(x,u,\frac{\partial V^{\pi^k}}{\partial x^{\top}}\right)\\ \text{s.t. }\ h(x(\tau),u(\tau))\geq 0\end{aligned} \tag{17}$$

It is worthwhile to mention that, different from the PEV, the PIM must be constrained in order to search for a feasible next policy. Therefore, it is necessary to reformulate a constrained PIM.

To cope with the constraints in the PIM stage, the soft constraints are introduced with the penalty function method. It approximates a constrained optimization problem with an unconstrained counterpart. The basic idea is to prescribe an additional cost penalty for constraint violation. In other words, the satisfaction of the safety constraints is achieved by optimizing a penalized objective added to the overall cost function. Hence, it is converted to optimizing the soft-constrained problem

$$\begin{aligned}\pi^{k+1}(x,t)=\arg\min_{\pi}(H\left(x,u,\frac{\partial V^{\pi^k}}{\partial x^{\top}}\right)+\rho\int_{t}^{T}\psi\mathrm{d}\tau)\\ \text{s.t. }\ \psi=\max\{0,-h(x(\tau),\pi(x(\tau),\tau))\}^2\end{aligned} \tag{18}$$

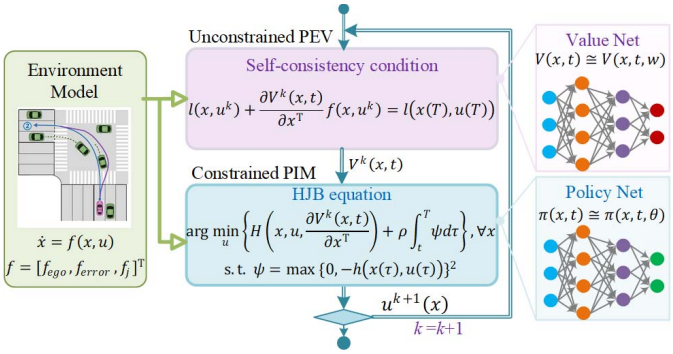

Fig. 3. Framework of the constrained ADP algorithm.

where the penalty term $\psi(\cdot)$ is correlated with the safety constraints, and $\rho > 0$ is the penalty coefficient. Once the constraints exceed the threshold, the penalty related to the constraints is enforced.

When we practically perform the policy iteration process (shown in Fig. 3), the integral terms ζ^π in (16) is omitted. It makes sense because ζ^π is small with a near-optimal policy. Considering (14), when $k \to \infty$, $\mathrm{d}H\left(x, u^{\pi^\infty}, \frac{\partial V^{\pi^\infty}}{\partial x}\right)/\mathrm{d}u = 0$, then $\zeta^{\pi^\infty} = 0$.

B. Parameterized Function for the Constrained ADP Algorithm

The generalization technique is usually utilized in ADP algorithm to approach the near-optimal solution. In this paper, we parameterize the value and control policy as neural networks. The parameterized value is also called "critic" with unknown parameter w and denoted as $V(x,t;w)$. Meanwhile, the parameterized control policy is called "actor" with unknown parameter θ, which is denoted as $u=\pi(x,t;\theta)$.

Fig. 3 is the flowchart of the parameterized constrained ADP algorithm. Both critic and actor are represented by deep neural networks for better accuracy of the approximated functions, which map from state-time pair to the near-optimal value function and control inputs. In order to the brevity, V_{w^k} is the abbreviation for $V(x,t;w^k)$.

At k-th step, PEV stage is carried out, which aims to solve the value V_{w^k} with respect to the known policy π^k. Considering the approximated solution to (16), one can reformulate it into minimizing J_{Critic}

$$J_{\text{Critic}} = \mathop{\mathbb{E}}_{x,t\sim d_{x,t}}\left\{\frac{1}{2}\left[H(x,\pi,\frac{\partial V_{w^k}}{\partial x^\top}) - l(x^\pi(T),u(T))\right]^2\right\}.$$

The mean value is calculated on the distribution of states and time. Its gradient is equal to

$$\frac{\partial J_{\text{Critic}}}{\partial w} = \mathop{\mathbb{E}}_{x,t\sim d_{x,t}}\left\{(H - l(x^\pi(T),u(T))\frac{\partial\left(\frac{\partial V_{w^k}}{\partial x^\top}f\right)}{\partial w}\right\}. \quad (19)$$

The critic parameter is tuned with the following updating rule

$$w^k = w^k - \alpha_w\frac{\partial J_{\text{Critic}}}{\partial w}, \quad (20)$$

where α_w indicates the size of each step.

Algorithm 1 Constrained ADP Algorithm for Dynamic Optimal Tracking of the Ego Vehicle

Initialize value and policy network $V^0(x,t;w^0)$, $\pi^0(x,t;\theta^0)$
Initialize the learning rates α_w and α_θ
Choose a small positive ϵ
repeat
  // Sampling
  Randomly select a reference path
  Initialize ego states x_{ego}, surrounding vehicles states x_j
  repeat
  Calculate control input using policy $u(t)=\pi_{\theta^k}(x(t),t)$
  Step to the next systems states $x(t)$ using (3) and (5).
  // Optimizing
  Rollout with policy π_{θ^k} from $\forall x_t \in \Omega$ to x_T
  1. Unconstrained PEV: Calculate $\frac{\partial J_{\text{Critic}}}{\partial w}$ by (19)
    Update value network $V^{\pi^k}(x,t;w)$ utilizing (20)
  2. Constrained PIM: Calculate $\frac{\partial J_{\text{Actor}}}{\partial \theta}$ by (23)
    Update policy $\pi^{k+1}(x,t;\theta)$ for all x with(24)
until $\|V_{w_{k+1}} - V_{w_k}\| \le \epsilon$

According to (18), the policy neural network aims to minimize the parameterized Hamiltonian

$$J_{\text{Actor}} = \mathop{\mathbb{E}}_{x,t\sim d_{x,t}}\left\{l(x,\pi(x,t;\theta^k)) + \frac{\partial V_{w^k}}{\partial x^\top}f(x,\pi(x,t;\theta^k)) + \rho\int_t^T\psi\mathrm{d}\tau\right\}. \quad (21)$$

where $\psi = \max\{0,-h(x(\tau),u(\tau))\}^2$, V_{w^k} is known and the parameters θ in $\pi(x,t;\theta^k)$ need to be solved. We define a penalty loss to avoid the collisions

$$J_{\text{Penalty}} = \mathop{\mathbb{E}}_{x,t\sim d_{x,t}}\left\{\int_t^T\psi\mathrm{d}\tau\right\} \quad (22)$$

To update the actor, one does the derivation the gradient of the actor loss concerning the parameter θ as

$$\frac{\partial J_{\text{Actor}}}{\partial\theta} = \mathop{\mathbb{E}}_{x,t\sim d_{x,t}}\left\{\left(\frac{\partial l(x,u)}{\partial u} + \frac{\partial\left(\frac{\partial V_{w^k}}{\partial x^\top}f\right)}{\partial u} + \rho\int_t^T\psi\mathrm{d}\tau\right)\frac{\partial\pi(x,t;\theta^k)}{\partial\theta}\right\}. \quad (23)$$

Afterwards, the actor is updated according to

$$\theta^k = \theta^k - \alpha_\theta\frac{\partial J_{\text{Actor}}}{\partial\theta} \quad (24)$$

where the step size α_θ is set by experience. Both the actor and critic are updated using gradient descent method in terms of their respective loss functions. The pseudo-code is shown in Algorithm 1 and Fig. 3 illustrates the design of the constrained ADP algorithm.

VI. Simulation Results

A. Descriptions of Driving Scenarios and Tasks

In this simulation, we choose the scenario as an intersection with three-way streets to verify the motion planning algorithm utilizing the constrained ADP algorithm, and three tasks are involved, i.e., left turn, going straight, and right turn. In the simulation, the length of the intersection is 50 m and the width

of the lane is 3.75 m in each direction of the intersection including turning left, going straight, and turning right. The ego vehicle is controlled by the proposed algorithm and the maneuver of surrounding vehicles is controlled by SUMO with the car-following model, lane-changing model and tracking routes. The traffic flow is 600 vehicles per hour on each lane. These three tasks are trained offline separately. In terms of the initial training data set, we select a batch of random system states. The ego vehicle's position p_x, p_y and heading angle ϕ are randomly selected from the reference points, and meanwhile the initial longitudinal velocity v_x is randomly set around the expected velocity. In addition, other states of the ego vehicle like yaw rate r and lateral velocity v_y are also set randomly. Meanwhile, the tracking errors are obtained according to the position of the ego vehicle and its closest point on the reference. Initially, the states of surrounding vehicles are randomly generated in SUMO, including the depart position and depart velocity.

For the first part of the integrated decision and control framework for motion planning, i.e., the path planning task, three path candidates are generated by the cubic Bezier curve. The key elements of the desired reference (R_x, R_y, R_φ) are given. As for the path-tracking controller, the parameters in the optimal control problem formulation is further set. Here, we choose the fixed final time T as 2.5 s. The choice of the time horizon is a balance of performance and computational effort. Meanwhile, the weighting parameters of position error, velocity error, steering input, and acceleration are designed as $q_1 = 0.5, q_2 = 0.8, r_1 = 0.5, r_2 = 0.5$. Assigning a relatively larger q_2 can prevent the vehicle from obtaining the policy of stopping moving (in order to avoid collisions). The output of the velocity of the vehicle as close as possible to the desired velocity to make the improvement on the traffic efficiency and safety. Meanwhile, Table I shows the detailed vehicle parameters. We use the same parameter settings with [29] except for the frequency. Although it is a continuous-time dynamic model, for simulation, it applies the discrete formulation as [31]. The radius of the vehicles are set as $r_{\text{veh}} = r_{\text{ego}} = 1$ m. To ensure the clearance of collision and also avoid over-conservatism (the free space could be reduced if the ego vehicle keeps a far distance from the surrounding vehicle under a larger safe distance), the parameter d_{safe} is set to be $r_{\text{veh}} + r_{\text{ego}} < d_{\text{safe}} < 2(r_{\text{veh}} + r_{\text{ego}})$. In this sense, we design the safe distance as a constant value, i.e., $d_{\text{safe}} = 3.5$ m. Moreover, we consider eight surrounding vehicles. Hence the dimensionality of the state is 41.

B. Details of the Algorithm

The constrained ADP algorithm is trained offline in an asynchronous learning architecture named ray [32]. To parameterize the value and policy, three-layer neural networks are adopted, which are fully connected and with 256 neurons in each hidden layer. The state-time pair is chosen as the input of both neural networks. Meanwhile, the activation function for the hidden layer and the output lay are exponential linear units (ELUs) and soft plus, respectively.

TABLE I
PARAMETERS OF THE EGO VEHICLE

Symbol	Value	Symbol	Value
k_1	-88000 [N/rad]	k_2	-94000 [N/rad]
m	1500 [kg]	a	1.14 [m]
b	1.40 [m]	I_{zz}	2420 [kg·m^2]
v_x	15 [m/s]	f	10 [Hz]

TABLE II
HYPERPARAMETERS OF THE CONSTRAINED ADP ALGORITHM

Hyperparameters	Value
Number of hidden layers	2
Number of neurons per hidden layer	256
Critic and Actor learning rates	Linear Annealing: 8e-5→ 8e-6
Number of Actors	2
Number of Learner	4
Penalty coefficient ρ	0.1
Total iteration	1×10^4
Batch size	512

Meanwhile, the activation function tanh is selected as the output of the policy neural network, in which case the policy is normalized to $[-1, 1]$. We further scale the angle of the front wheel to $[-0.4, 0.4]$ rad while shifting the acceleration to $[-3, 1.5]$m/s^2. To accomplish the updating both neural networks, the Adam optimization method is leveraged. The specific hyperparameters are designed as shown in Table II. The number of actors and learners relies on the number of CPU cores under the framework of Ray. This high-throughput asynchronous learning architecture handles the modules of buffer, learner, evaluator optimizer and actor in parallel. Then, the batch size is designed as 512 to estimate the gradient of actor and critic. As soon as the ego vehicle passes the intersection without colliding with obstacles or violating traffic laws, the training ends. The neural networks are trained for 1×10^3 iterations on a single trail, and there are five different training trails. Using different random seeds ensures diversity among trails.

C. Analysis of the Results

Utilizing the constrained ADP method in Algorithm 1, the obtained policy neural network is applied online as a mapping function from the states to control input. It is noted that this policy neural network has been trained offline to solve the constrained optimal control problem (including the vehicle stability and safety constraints), which holds for all states $x \in \Omega$ and aims to track the desired trajectory safely. The offline training determines the performance of the policy. Hence, we demonstrate the tracking and safety performance of different tasks during the training process in Fig. 4. In Fig. 4(a), the value loss J_{Critic} reflects the training performance of the value function, which influences the choice of the reference path. Fig. 4(b) and Fig. 4(c) demonstrate the training performance of the policy loss J_{Actor} and the safety penalty J_{Penalty}. As shown in Fig. 4, the learning curves of the value loss, policy loss, and safety penalty decrease consistently for these three scenarios along the training process, which indicates that the policy

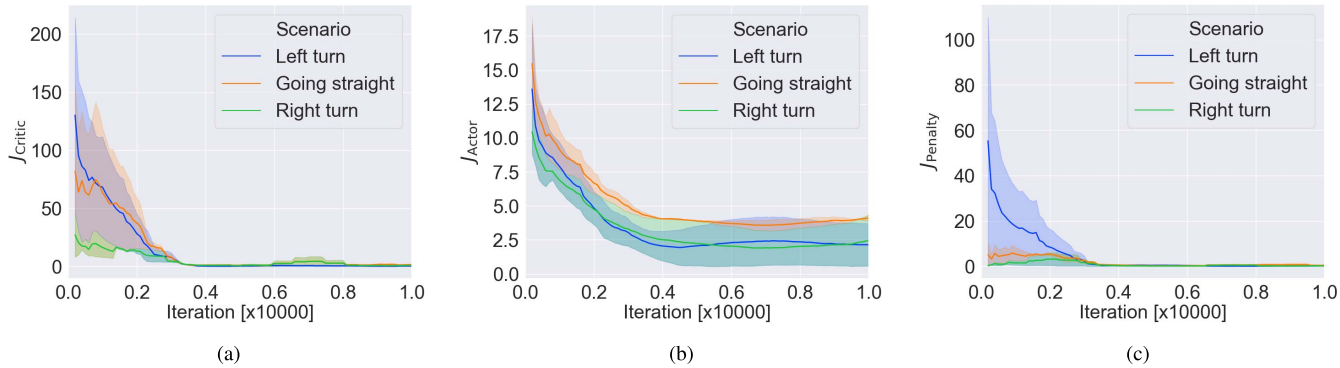

Fig. 4. The loss performance of optimal control in the offline training process. (a) Value loss performance (b) Policy loss performance (c) Penalty loss at the training stage. The mean is painted by the solid line and the shaded regions reflect 95% confidence interval for 5 trials.

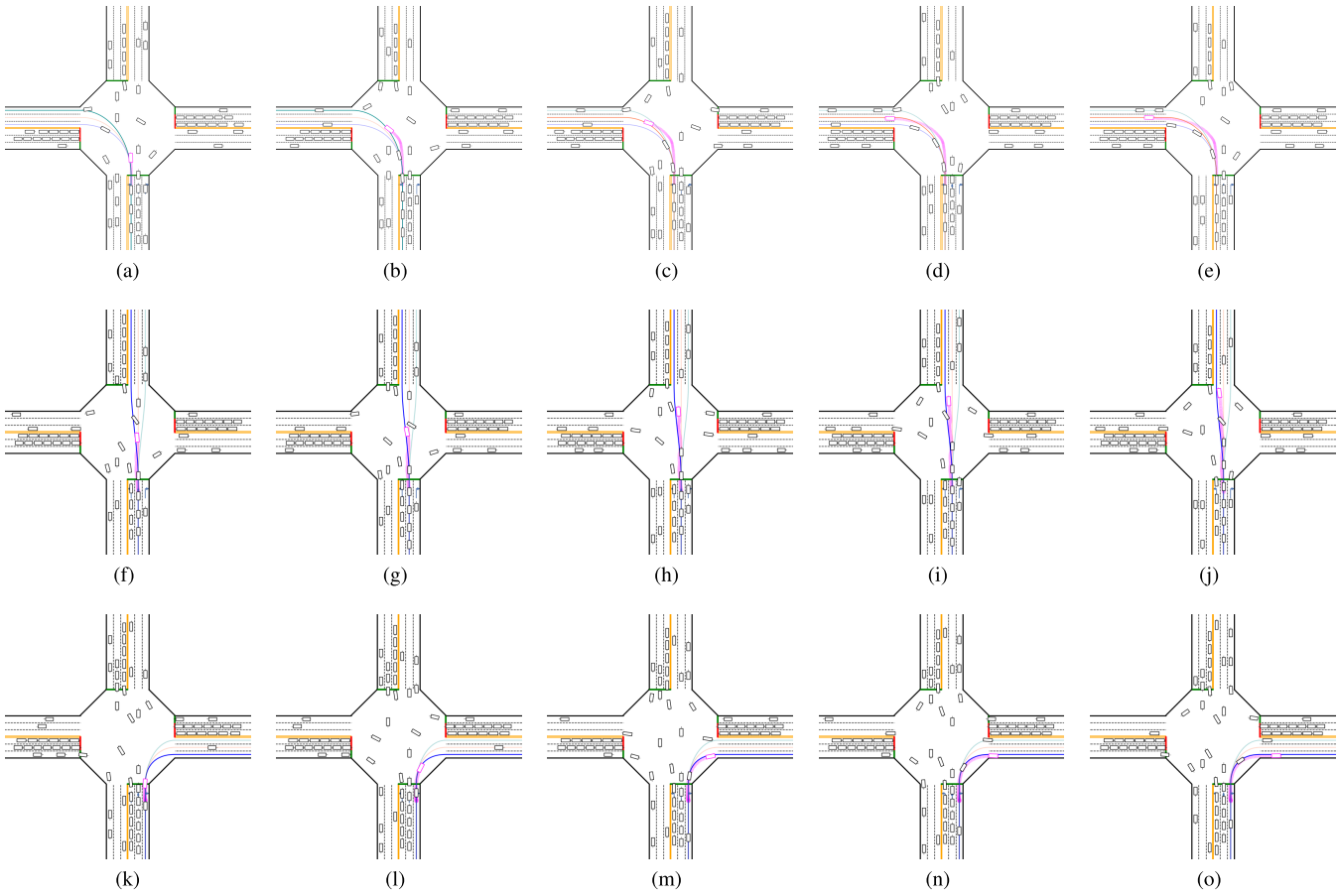

Fig. 5. Tracking performance of the left-turn, going straight, and right-turn driving scenarios.

obtained by the algorithm is convergent to the optimality. After 1×10^3 iterations, both value loss and the safety penalty decrease to approximately zero, which indicates that the algorithm is convergent in the training process.

The optimal policy is applied online at the intersection during simulation, which is learned from 1×10^4 iterations in the training process utilizing the proposed constrained ADP algorithm. We set $t = 0$ and apply the learned control input $\pi(x, 0; \theta^*)$ in the form of the rolling horizon. Fig. 5 shows the tracking performance when the autonomous vehicle follows the given policy in left-turn, going straight and right-turn scenarios. The results show that the policy can achieve the task of tracking the desired trajectory and avoiding collisions successfully. In addition, the optimal tracking path is adjusted dynamically and it is chosen from the fixed candidate paths according to the value network during the testing process. The detailed performance of these three tasks is analyzed as follows. For the task of turning left, at the beginning, the autonomous vehicle chooses the top path as the initial reference path shown in Fig. 5(a) and Fig. 5(b). Then, to avoid collision with surrounding vehicles and minimize the tracking errors, it changes to select the middle path as the reference,

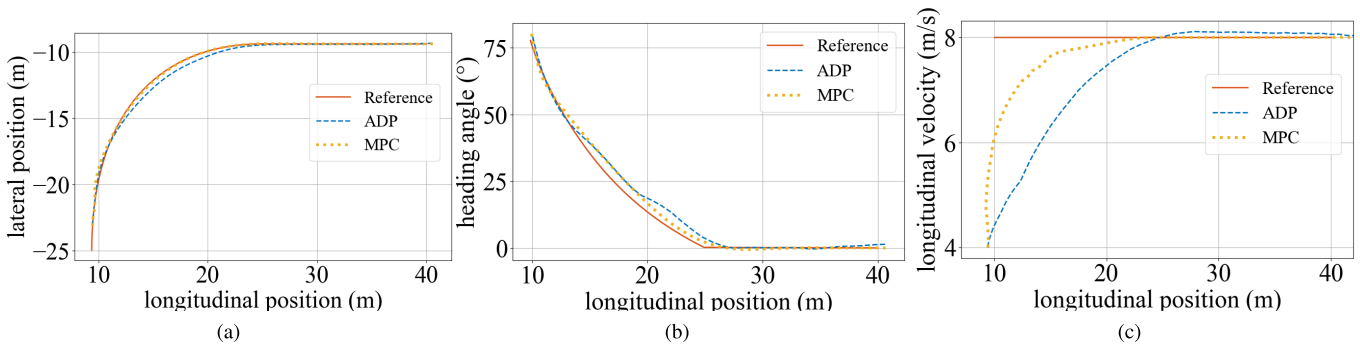

Fig. 6. Tracking performance comparison between the proposed constrained ADP algorithm and MPC for the right-turn scenario.

as shown in Fig. 5(c). Finally, it achieves the task of turning left safely as shown in Fig. 5(d) and it exits from the middle path as depicted in Fig. 5(e). As for the task of going straight shown in Fig. 5(f) and Fig. 5(g), when the autonomous vehicle meets the danger of running into the surrounding vehicle as shown in Fig. 5(h), it is controlled to steer the front wheel to avoid the collision as shown in Fig. 5(i) and 5(j). Finally, it proceeds to go straight through the intersection. As for the right-turn scenarios, the controller also leads to a satisfying performance as indicated in Fig. 5(k)-Fig. 5(o).

To evaluate the effectiveness of the constrained ADP algorithm, the MPC method is utilized as a baseline. It approximates the optimal solution, though it solves the discrete-time optimal control problem. Similar to (13), the MPC problem is formulated as

$$\begin{aligned} \min_{u_t,\cdots,u_{t+T-1}\in U} V(x,t) &= \sum_{n=0}^{N}(l(x(n|t),u(n|t)) \\ \text{s.t. } x(n+1|t) &= x(n|t)+f(x(n|t),u(n|t))\Delta t, \\ h(x(n|t),u(n|t)) &\geq 0, \quad n=0:N-1 \end{aligned} \tag{25}$$

where

$$\begin{aligned} &l(x(n|t),u(n|t)) \\ &= q_1\left\|M_d x(n|t)\right\|^2 + q_2\left\|M_v x(n|t)\right\|^2 \\ &\quad + r_1 u(n|t)^T M_\delta u(n|t) + r_2 u(n|t)^T M_a u(n|t), \end{aligned} \tag{26}$$

$\Delta t = 0.1$ and $N = 25$. This design aims to be consistent with the continuous-time ADP problem. Meanwhile, the weighting parameters are equivalent to the ones in ADP problem. To obtain the solution of the MPC problem (25), with Casadi package, the IPOPT solver with [33] is utilized in this study.

For demonstrative purposes, we pick the right-turn scenario to demonstrate the effectiveness of the proposed ADP methodology through comparisons. First of all, we compare one-step online computation times for ADP and MPC on a PC with an i7-8850H processor. It shows that the computational time of our method is 0.2 ms averagely, while nonlinear MPC needs nearly 100 ms for this constrained optimization problem. It is concluded that the proposed constrained ADP method owns higher efficiency than the nonlinear MPC method.

Besides the calculation time, we also analyze the performance in this driving scenario. Fig. 6 depicts the performance attained by the proposed constrained ADP algorithm and the MPC. Note that the solutions of MPC can be usually regarded as the nearly optimal solution, and obviously the curves of state and control by using ADP algorithm 1 align well with those of the MPC. The precision of the constrained ADP compared with MPC is verified in the test. Fig. 6(a) displays the trajectory of the lateral position, whose maximum difference is around 0.4 m for ADP while 0.2 m for MPC. The heading angle curve is shown as Fig. 6(b), where both maximum errors are no more than 7°. Fig. 6(c) reflects the longitudinal velocity, where the final longitudinal velocity error for the constrained ADP algorithm is less than 1 m/s. Therefore, the tracking performance of constrained ADP algorithm is comparable to the one of MPC (which demonstrates the optimality), but the computational burden is greatly relieved.

VII. Conclusion

This work presents a constrained ADP method for autonomous driving in the constrained dynamic environment. The framework of the integrated decision and control framework is utilized, which achieves planning static paths and tracking optimal path. The optimal desired reference path is chosen among all the path candidates that are generated in advance. To track the trajectory, we formulate the optimal control problem and propose the constrained ADP algorithm. Utilizing the actor-critic framework, value and policy neural networks with multi-layers are deployed online for path selection and tracking. Finally, a series of simulations are conducted based on three driving scenarios, i.e., left-turn, going straight, and right-turn, which show the generality and efficiency of the constrained ADP algorithm. They indicate that the proposed ADP algorithm performs effectively and safely in these driving scenarios. With the generalized policy, the proposed approach is suitable for dynamically varying driving scenarios with changing surrounding vehicle constraints. Meanwhile, the system performance of our algorithm is comparable to that of MPC, and the computational efficiency of the proposed method is significantly improved. In the future, we will cope with the problem of different dimensions of policy inputs and then train a single policy network suitable for different scenarios.

REFERENCES

[1] L. D. Burns, "A vision of our transport future," *Nature*, vol. 497, no. 7448, pp. 181–182, May 2013.

[2] J. Duan, Z. Liu, S. E. Li, Q. Sun, Z. Jia, and B. Cheng, "Adaptive dynamic programming for nonaffine nonlinear optimal control problem with state constraints," *Neurocomputing*, vol. 484, pp. 128–141, May 2022.

[3] 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, Oct. 2010.

[4] F. Borrelli, P. Falcone, T. Keviczky, J. Asgari, and D. Hrovat, "MPC-based approach to active steering for autonomous vehicle systems," *Int. J. Veh. Auto. Syst.*, vol. 3, nos. 2–4, pp. 265–291, 2005.

[5] S. M. Erlien, S. Fujita, and J. C. Gerdes, "Shared steering control using safe envelopes for obstacle avoidance and vehicle stability," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 2, pp. 441–451, Feb. 2016.

[6] E. Siampis, E. Velenis, S. Gariuolo, and S. Longo, "A real-time nonlinear model predictive control strategy for stabilization of an electric vehicle at the limits of handling," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 6, pp. 1982–1994, Nov. 2017.

[7] S. Dixit et al., "Trajectory planning for autonomous high-speed overtaking in structured environments using robust MPC," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 6, pp. 2310–2323, Jun. 2019.

[8] P. Hang, C. Lv, Y. Xing, C. Huang, and Z. Hu, "Human-like decision making for autonomous driving: A noncooperative game theoretic approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2076–2087, Apr. 2021.

[9] Z. Liu et al., "Recurrent model predictive control: Learning an explicit recurrent controller for nonlinear systems," *IEEE Trans. Ind. Electron.*, vol. 69, no. 10, pp. 10437–10446, Oct. 2022.

[10] 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.

[11] J. Ma, Z. Cheng, X. Zhang, M. Tomizuka, and T. H. Lee, "Alternating direction method of multipliers for constrained iterative LQR in autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 12, pp. 23031–23042, Dec. 2022.

[12] X. Zhang, Z. Cheng, J. Ma, S. Huang, F. L. Lewis, and T. H. Lee, "Semi-definite relaxation-based ADMM for cooperative planning and control of connected autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 9240–9251, Jul. 2022.

[13] J. Ma, Z. Cheng, X. Zhang, Z. Lin, F. L. Lewis, and T. H. Lee, "Local learning enabled iterative linear quadratic regulator for constrained trajectory planning," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, May 2, 2022, doi: 10.1109/TNNLS.2022.3165846.

[14] 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.

[15] C. Mu and Y. Zhang, "Learning-based robust tracking control of quadrotor with time-varying and coupling uncertainties," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 1, pp. 259–273, Jan. 2020.

[16] J. Duan, Y. Guan, S. E. Li, Y. Ren, Q. Sun, and B. Cheng, "Distributional soft actor-critic: Off-policy reinforcement learning for addressing value estimation errors," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 11, pp. 6584–6598, Nov. 2022.

[17] K. Makantasis, M. Kontorinaki, and I. Nikolos, "Deep reinforcement-learning-based driving policy for autonomous road vehicles," *IET Intell. Transp. Syst.*, vol. 14, no. 1, pp. 13–24, Jan. 2020.

[18] Z. Lin et al., "Solving finite-horizon HJB for optimal control of continuous-time systems," in *Proc. Int. Conf. Comput., Control Robot. (ICCCR)*, Jan. 2021, pp. 116–122.

[19] S. E. Li, *Reinforcement Learning for Sequential Decision and Optimal Control*. Singapore: Springer-Verlag, 2023.

[20] J. Chen, S. E. Li, and M. Tomizuka, "Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 5068–5078, Jun. 2022.

[21] P. Werbos, "Approximate dynamic programming for real time control and neural modelling," in *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*. New York, NY, USA: Van Nostrand Reinhold, 1992, pp. 493–525.

[22] W. B. Powell, *Approximate Dynamic Programming: Solving The Curses of Dimensionality*. Hoboken, NJ, USA: Wiley, 2007.

[23] R. A. Howard, *Dynamic Programming and Markov Processes*. Cambridge, MA, USA: MIT Press, 1960.

[24] Z. Lin, J. Duan, S. E. Li, H. Ma, Y. Yin, and B. Cheng, "Continuous-time finite-horizon ADP for automated vehicle controller design with high efficiency," in *Proc. 3rd Int. Conf. Unmanned Syst. (ICUS)*, Nov. 2020, pp. 978–984.

[25] D. Liu, Q. Wei, D. Wang, X. Yang, and H. Li, *Adaptive Dynamic Programming With Applications in Optimal Control*. London, U.K.: Springer, 2017.

[26] Z. Zhao, Y. Yang, H. Li, and D. Liu, "Approximate finite-horizon optimal control with policy iteration," in *Proc. 33rd Chin. Control Conf.*, Jul. 2014, pp. 8895–8900.

[27] J. Zhao, "Optimization based on motion planning of nonholonomic nonlinear systems using global adaptive dynamic programming," in *Proc. 31st Youth Academic Annu. Conf. Chin. Assoc. Autom. (YAC)*, Nov. 2016, pp. 433–438.

[28] P. J. Werbos, "Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-17, no. 1, pp. 7–20, Jan. 1987.

[29] Z. Lin et al., "Policy-iteration-based finite-horizon approximate dynamic programming for continuous-time nonlinear optimal control," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Dec. 5, 2022, doi: 10.1109/TNNLS.2022.3225090.

[30] Y. Guan et al., "Integrated decision and control: Toward interpretable and computationally efficient driving intelligence," *IEEE Trans. Cybern.*, vol. 53, no. 2, pp. 859–873, Feb. 2023.

[31] 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. Vehicles Symp. Workshops (IV Workshops)*, Jul. 2021, pp. 128–134.

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

[33] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, Mar. 2018.

Ziyu Lin received the B.S. degree in automotive engineering from China Agricultural University, Beijing, China, in 2017. She is currently pursuing the Ph.D. degree with the School of Vehicle and Mobility, Tsinghua University, Beijing. She studied as a Visiting Student Researcher with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore, from 2021 to 2022. She was a recipient of the Best Paper Award at the IEEE 2020 3rd International Conference on Unmanned Systems and the Best Presentation Award on IEEE 2021 International Conference on Computer Control and Robotics. Her current research interests include reinforcement learning, approximate dynamic programming, model predictive control, and autonomous driving.

Jun Ma received the B.Eng. degree (Hons.) in electrical and electronic engineering from Nanyang Technological University, Singapore, in 2014, and the Ph.D. degree in electrical and computer engineering from the National University of Singapore, Singapore, in 2018. From 2018 to 2021, he held several positions at the National University of Singapore; University College London, London, U.K.; University of California at Berkeley, Berkeley, CA, USA; and Harvard University, Cambridge, MA, USA. He is currently an Assistant Professor with the Robotics and Autonomous Systems Thrust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China, an Assistant Professor with the Division of Emerging Interdisciplinary Areas, and an Affiliate Assistant Professor with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong SAR, China. His research interests include control, optimization, and machine learning with application to robotics, and autonomous driving.

Jingliang Duan (Member, IEEE) received the Ph.D. degree from the School of Vehicle and Mobility, Tsinghua University, China, in 2021. He studied as a Visiting Student Researcher with the Department of Mechanical Engineering, University of California at Berkeley, Berkeley, in 2019, and worked as a Research Fellow with the Department of Electrical and Computer Engineering, National University of Singapore, from 2021 to 2022. He is currently an Associate Professor with the School of Mechanical Engineering, University of Science and Technology Beijing, China. His research interests include reinforcement learning, optimal control, and self-driving decision-making.

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

He worked at Stanford University, University of Michigan, and University of California at Berkeley, Berkeley. He is currently a Tenured Professor at Tsinghua University. His active research interests include intelligent vehicles and driver assistance, reinforcement learning, distributed control, optimal control, and estimation. He is the author of over 100 journals/conference papers and the co-inventor of over 20 Chinese patents. He was a recipient of the Best Paper Award in 2014 IEEE ITS Symposium, Best Paper Award in 14th ITS Asia Pacific Forum, National Award for Technological Invention in China in 2013, Excellent Young Scholar of NSF China in 2016, and 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 received the B.S. and master's degrees from the School of Vehicle and Mobility, Tsinghua University, Beijing, China, in 2019 and 2022, respectively. He is currently pursuing the Ph.D. degree with Harvard University. His current research interests include optimal control and safe-critical systems.

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 the Tsinghua University Suzhou Automotive Research Institute. He is the author of more than 100 peer-reviewed journals/conference papers and the co-inventor of 40 patents. His active research interests include autonomous vehicles, driver-assistance systems, active safety, and vehicular ergonomics, among others. He 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 National 863 Plan, among others.

Tong Heng Lee received the B.A. degree (Hons.) in electrical engineering from the University of Cambridge, Cambridge, U.K., in 1980, the M.Eng. degree in electrical engineering from the National University of Singapore (NUS), Singapore, in 1985, and the Ph.D. degree in electrical engineering from Yale University, New Haven, CT, USA, in 1987.

He is currently a Professor with the Department of Electrical and Computer Engineering, National University of Singapore, and also a Professor with the NUS Graduate School, NUS NGS. He was the Past Vice-President (Research) of NUS. His research interests include the areas of adaptive systems, knowledge-based control, intelligent mechatronics, and computational intelligence.

Prof. Lee currently holds Associate Editor appointments in the IEEE TRANSACTIONS ON CYBERNETICS, *Control Engineering Practice* (IFAC), and the *International Journal of Systems Science* (Taylor and Francis, London). In addition, he is the Past Deputy Editor-in-Chief of *Mechatronics* (IFAC) journal.