IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 11, 2020, accepted October 21, 2020, date of publication October 27, 2020,
date of current version December 3, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.3034225*

# Hierarchical Reinforcement Learning for Autonomous Decision Making and Motion Planning of Intelligent Vehicles

**YANG LU[1], XIN XU[1], (Senior Member, IEEE), XINGLONG ZHANG[1], (Member, IEEE), LILIN QIAN[2], AND XING ZHOU[1]**

[1]College of Intelligence Science and Technology, National University of Defense Technology, Changsha 410073, China
[2]Unmanned System Technology Research Center, National Innovation Institute of Defense Technology, Beijing 100000, China

Corresponding author: Xin Xu (xinxu@nudt.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61751311 and Grant 61825305, and in part by the National Key Research and Development Program of China under Grant 2018YFB1305105.

**ABSTRACT** Autonomous decision making and motion planning in complex dynamic traffic environments, such as left-turn without traffic signals and multi-lane merging from side-ways, are still challenging tasks for intelligent vehicles. It is difficult to generate optimized behavior decisions while considering the motion capabilities and dynamic properties of intelligent vehicles. Aiming at the above problems, this article proposes a hierarchical reinforcement learning approach for autonomous decision making and motion planning in complex dynamic traffic scenarios. The proposed approach consists of two layers. At the higher layer, a kernel-based least-squares policy iteration algorithm with uneven sampling and pooling strategy (USP-KLSPI) is presented for solving the decision-making problems. The motion capabilities of the ego vehicle and the surrounding vehicles are evaluated with a high-fidelity dynamic model in the decision-making layer. By doing so, the consistency between the decisions generated at the higher layer and the operations in the lower planning layer can be well guaranteed. The lower layer addresses the motion-planning problem in the lateral direction using a dual heuristic programming (DHP) algorithm learned in a batch-mode manner, while the velocity profile in the longitudinal direction is inherited from the higher layer. Extensive simulations are conducted in complex traffic conditions including left-turn without traffic signals and multi-lane merging from side-ways scenarios. The results demonstrate the effectiveness and efficiency of the proposed approach in realizing optimized decision making and motion planning in complex environments.

**INDEX TERMS** Autonomous driving, hierarchical reinforcement learning, complex dynamic traffics, decision making, motion planning.

## I. INTRODUCTION

Autonomous driving technology (ADT) has received extensive attention in the past decade and has made much progress. ADT is an important part of the future intelligent transport system which is promising to realize transportation safety, efficiency and energy conservation. However, in complex traffic conditions such as left-turn without traffic signals and multi-lane merging from side-ways etc., there are still many challenges in achieving fully autonomous driving [1]. Among them, behavioral decision making and motion planning for solving complex dynamic traffic scenarios are two major challenges.

In a typical decision-making and motion-planning system, the decision-making module is in charge of generating high-level orders, such as slow down and speed up; while the motion planner is to compute the detailed trajectory profile that can be followed by the vehicle using low-level controllers. There have been fruitful works contributed to improving the decision-making behaviors in complex environment and the motion planning performance in terms of mobility and smoothness, see for instance [2], [3]. The above works addressed the decision-making or motion planning problems respectively. In fact, the performance of the

The associate editor coordinating the review of this manuscript and approving it for publication was Michail Makridis.

decision-making and motion planning are mutually coupled. As a specific example, the decision-making, if not considering the motion capability of the vehicle, might lead to a policy that can not respect the limit of vehicle dynamics. With this information, the trajectory computed by the motion planner might not be reachable to the low-level controllers, which leads to inconsistency issues.

In this article, we develop a hierarchical structure based on reinforcement learning to address this issue. In contrast to rule-based approaches that might generate conflict decisions in complex environment, we propose a reinforcement learning-based algorithm for making decisions in continuous state space. Also, a high-fidelity dynamical model is utilized for collecting state samples of the ego and surrounding vehicles, which are then used for generating consistent decision-making policies via off-line training. Note that, in [4], a time-efficient motion planning algorithm was developed for maneuver in urban scenarios. The difference our paper lies in the following two aspects: i) the decision-making and motion planning policies are learned and improved in a fully data-driven way; ii) the online computational issue is not a major concern since the policy can be learned off-line in a batch-mode manner.

Hence, the main contributions can be summarized as follows:

- We propose a hierarchical reinforcement learning algorithm for decision making and motion planning of intelligent vehicles. At the higher layer, a sample-efficient kernel-based least-squares policy iteration algorithm with uneven sampling and pooling strategy (USP-KLSPI) is presented for decision making, while the motion planner in the lower layer utilizes a batch-mode dual heuristic programming (DHP) to generate the trajectory in the lateral direction.
- The motion capabilities of the ego vehicle and the surrounding vehicles are considered at the higher layer in the training process using a high-fidelity vehicle dynamic model. In this way, the consistency between the decisions generated at the higher layer and the operations learned in the lower planning layer can be well guaranteed.
- Extensive simulations on decision making and motion planning are conducted in complex traffic conditions including left-turn without traffic signals and multi-lane merging from side-ways scenarios. The results demonstrate the effectiveness and efficiency of the proposed approach in realizing optimized decision making and motion planning.

The remainder of this article is arranged as follows. Related works and research background are given in Section II. The framework of HRL, decision making in two complex scenarios, as well as motion planning for avoiding obstacles are presented in section III, while Section IV provides simulation, analyses, and discussion. Finally, the conclusions and future work are drawn in Section V.

## II. RELATED WORKS AND RESEARCH BACKGROUND

### A. RELATED WORKS

In the aspect of decision making, rule-based decision-making methods were generally utilized in earlier research on intelligent vehicles. Among them, the most typical rule-based decision-making method is finite state machine (Finite State Machine, FSM) [5], [6]. Rule-based methods were widely applied in solving decision-making problems of intelligent vehicles because of the clear logic and strong interpretability. However, problems such as decision conflicts or discontinuous decisions may occur due to overlapping judgment conditions or discontinuous state divisions. Besides, rule-based decision-making methods rarely have the ability to deal with unseen scenarios.

In recent years, applications of machine learning methods in behavioral decision making have received much attention. Ngai *et al.* proposed a tabular reinforcement learning (RL) framework to learn lane-changing and overtaking behaviors [7]. The tabular Q-learning method converges slowly. In addition, it has computational and storage burdens in behavioral decisions of continuous space. In [8], kernel-based approximate policy iteration (API) algorithm was utilized to deal with the overtaking problems in the highway. Compared with traditional tabular RL methods in solving decision-making problems, it improves the learning efficiency. In [9], the extreme learning machine (ELM) was introduced to the batch-mode RL for realizing fast and efficient feature reconstruction, and Liu *et al.* verified its effectiveness in the lane-changing decision-making problem. Kernel-based API in [8] and ELM-API in [9] dealt with simple overtaking tasks and lane-changing issues without considering the trajectory planning when there exists the obstacles. In [10]–[14], deep learning approaches were used for training the collected raw sensor samples in specific scenarios which can map the images to actions. The advantages of using deep neural networks include automatic feature extraction and feature representation. However, the complex dynamic scenarios such as left-turn without traffic signals and multi-lane merging from side-ways were not considered in previous works. In addition, it is still difficult to generate optimized behavior decisions while considering the motion capabilities and dynamic properties of intelligent vehicles.

As a connection between behavior decision and lower-level control, the motion planning module of intelligent vehicles needs to plan kinematically or dynamically feasible trajectories for tracking control. Previous motion-planning methods can be classified into four categories [15]: graph search based [16]–[21], sampling based [22]–[24], interpolation curve [25]–[28], and numerical optimization [29]–[31]. In [29], Dolgov *et al.* proposed to use A* search to obtain a kinematically feasible trajectory and utilized numerical optimization method to smooth the previously calculated trajectory. The computation time of A* search method is expensive and is suitable for known unstructured low-speed scenarios. As a kind of sequential optimization decision approach,

RL has potential in solving the motion-planning problems of nonlinear systems. In [32], Lian *et al.* proposed the use of Heuristic Dynamic Programming (HDP) to avoid obstacles. Experimental tests on wheeled mobile robots verified the effectiveness of the method. In [33], Al Dabooni *et al.* proposed the Dyna-HDP algorithm for vehicle planning tasks which can find approximate optimal trajectories. HDP in [32] and Dyna-HDP in [33] were utilized to complete the motion-planning tasks. The above works are mainly concerned about the optimization in the planning layer. Nevertheless, since there is a close coupling relationship between behavioral decision making and motion planning, in complex dynamic scenarios, there may be conflicts between the two layers. In a previous work [34], Qian *et al.* proposed a decision-making algorithm using the planning feature in the training process for performance improvement. However, the planning feature used cannot completely represent the motion capability of the vehicles.

### B. RESEARCH BACKGROUND

Markov decision processes (MDPs) can provide efficient mathematical frameworks for RL. As the system model is unknown or partially known in complex dynamic scenarios, it is difficult to calculate an accurate solution to solve the optimal decision-making problems. In RL, approximate policy iteration and actor-critic algorithms have been studied to solve MDPs with unknown model and large state spaces.

#### 1) THE MDP MODEL FOR RL

An MDP model can be represented as a quaternion tuple $[S, A, P_{sa}, \mathcal{R}]$, where $S$ and $A$ are the sets of states and actions, respectively. $P_{sa}$ is the state transition probability from state $s_t$ to $s_{t+1}$ after performing an action $a_t$. When it occurs state transition from the state $s_t$ to the next state $s_{t+1}$ after taking an action $a_t \in A$, we can obtain the corresponding reward $\mathcal{R}(s_t, a_t)$, i.e. $S \times A \to \mathbb{R}$.

In RL, the state-action value function for a given MDP model is defined as follows:

$$Q^{\pi}(s_t, a_t) = E^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \,|s_0 = s_t, a_0 = a_t\right] \quad (1)$$

where $E^{\pi}(\cdot)$ is the mean of stationary policy $\pi$ which can map the state to the action space, that is $\pi : S \to A$. $r(s_t, a_t) \in \mathcal{R}$ is the reward, the $\gamma \in (0, 1]$ is the discount factor and the equation above can be described by

$$Q^{\pi}(s_t, a_t) = E^{\pi}[r(s_t, a_t) + \gamma \sum_{a_{t+1} \in A} P_{sa}(s_{t+1}, a_{t+1}) Q^{\pi}(s_{t+1}, a_{t+1})] \quad (2)$$

#### 2) APPROXIMATE POLICY ITERATION

In [35], Lagoudakis *et al.* considered the advantages of least-squares temporal-difference (LSTD) and approximate policy iteration (API). Then the least-squares policy iteration (LSPI) was proposed. LSPI uses a linear weighted combination of $k$ basis functions $\sum_{i=1}^{k} \phi_i(s_t, a_t)$ to approximate the state-action value function $Q^{\pi}$ [35].

$$\hat{Q}^{\pi}(s_t, a_t) = \sum_{i=1}^{k} \phi_i(s_t, a_t) w_i^{\pi} = \phi(s_t, a_t)^{\top} w^{\pi} \quad (3)$$

where $\phi(s_{t+1}, a_{t+1}) \in \mathbb{R}^{1\times k}$. Hence equation (2) can be rewritten in the format of matrix as follows:

$$\phi(s_t, a_t)^{\top} w^{\pi} = E[r(s_t, a_t) + \gamma \sum_{a_{t+1} \in A} P_{sa}^{\pi} \phi(s_{t+1}, a_{t+1})^{\top} w^{\pi}] \quad (4)$$

$a_{t+1} = \pi(s_{t+1})$. According to [36], the weighted least-squares solution would be:

$$w^{\pi} = \mathbf{A}^{-1}\mathbf{b} \quad (5)$$

According to [35], $\mathbf{A}$ and $\mathbf{b}$ can be approximated using collected samples. The state-action pair $(s_i, a_i)$ is denoted as $s_i$, and $(s_{i+1}, a_{i+1})$ is denoted as $s_{i+1}$ for convenience. Hence, the approximated value of $\mathbf{A}$ and $\mathbf{b}$ are written as:

$$\begin{aligned} \widehat{\mathbf{A}} &= \widehat{\mathbf{A}} + \sum_{i=1}^{T} \phi(s_i)^{\top} [\phi(s_i) - \gamma \phi(s_{i+1})] \\ \widehat{\mathbf{b}} &= \widehat{\mathbf{b}} + \sum_{i=1}^{T} \phi(s_i)^{\top} r_i \end{aligned} \quad (6)$$

$s_k = [s_i, a_i, s_{i+1}, r_i] (k = 1, \cdots, T)$, where $s_k$ represents the training samples and T is the number of samples. LSPI utilizes the greedy strategy to choose the action that can maximize the state-action value function $Q^{\pi}$.

To improve the learning efficiency and convergence speed, Xu *et al.* proposed a kernel-based least-squares policy iteration algorithm (KLSPI) which uses a technique of approximate linear dependence (ALD) [37]. The decision-making policy can be trained by KLSPI with the pre-collected samples. Due to the ability of non-linear feature representation, KLSPI has a superiority in dealing with decision-making problems of large-scale state and action spaces. In addition, when the number of samples is large, it can still learn the approximate optimal policy. Compared with LSPI, the KLSPI algorithm has better convergence properties and can obtain an approximate optimal solution.

#### 3) HRL FOR OPTIMIZING DECOMPOSED SUBTASKS

When intelligent vehicles drive in complex dynamic environments, the decision making and motion planning have to be optimized at the same time. One advantage of hierarchical reinforcement learning is solving complex tasks based on task decomposition. In [38], a hierarchical-DQN framework was proposed. The higher layer learns a policy over intrinsic goals and the lower layer learns a policy that meets the goal from the atomic actions. In [39], Ding *et al.* proposed a hierarchical framework of which the upper layer generates the control actions by hidden Markov model and the lower layer realizes the planning for agents' navigation by deep reinforcement

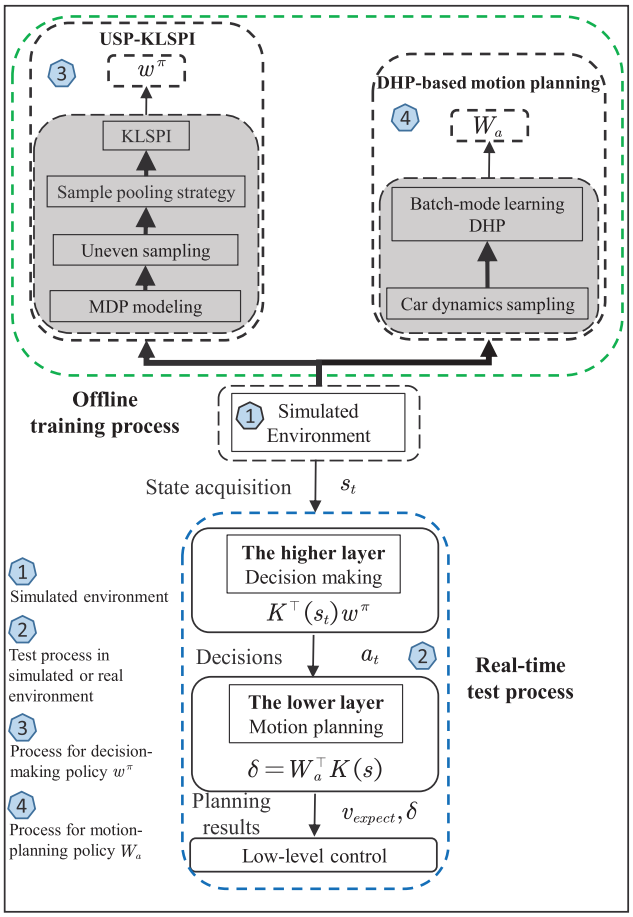



**FIGURE 1.** The framework of HRLDP: $s_t$ is the state perceived by the ego vehicle. $v_{expect}$ is longitudinal speed and $\delta$ is the front steering angle. The thick arrows indicate the transfer of batch data, and the thin arrows represent the transfer of matrix or several states.

learning. To the best of our knowledge, there is little work on hierarchical reinforcement learning for autonomous decision making and motion planning of intelligent vehicles in complex traffic conditions such as left-turn without traffic signals and multi-lane merging from side-ways.

## III. THE PROPOSED HRL APPROACH FOR OPTIMIZED DECISION MAKING AND MOTION PLANNING

From the task decomposition perspective, we propose a hierarchical reinforcement learning approach for solving decision-making and motion-planning problems, as depicted in Fig.1.

At the higher layer, we utilize the USP-KLSPI to make decisions. The process for obtaining decision-making policy includes four parts: MDP modeling of the decision-making tasks, uneven sampling, sample pooling strategy, and the KLSPI algorithm.

At the lower layer, the real-time obstacle avoidance is solved by the planning policy being trained via a DHP in a batch-mode way. Finally, the expected longitudinal speed $v_{expect}$ and the steering angle $\delta$ are transferred to the low-level controller.

In the both layers, the data samples used for training are collected using a high-fidelity 14-degree-of-freedom (DOF) dynamics, which is referred to the previous work in [8], [40]. The dynamics of the built vehicle consists of 3 modules, which are the steering system, the body suspension model, and the motion. Also, the effectiveness of the model to the real vehicle dynamics is verified.

The main steps of HRLDP is shown in Algorithm 1. The algorithm includes the higher decision-making layer and the lower motion-planning layer.

**Algorithm 1** Hierarchical Reinforcement Learning for Decision Making and Motion Planning of Intelligent Vehicles

\\ **The higher layer.**
**Require:** Scenario information, the trained decision-making policy $w^\pi$.
**Ensure:** The decision considering the longitudinal speed.
Obtain the states of the ego vehicle and the ruled-based surrounding vehicles, i.e. $s_t$;
Compute $\hat{Q}^\pi$ at the state $s_t$ by $K^\top(s_t)w^\pi$;
Generate the final decision by $\arg\max_{a_t} \hat{Q}^\pi(s_t, a_t)$;
\\ **The lower layer.**
**Require:** The decision containing the longitudinal speed information, the trained motion-planning policy $W_a$.
**Ensure:** Planning results to low-level control.
Obstacle detection and obtaining current vehicle state $s$;
Generate the front steering angle for lateral trajectory planning, i.e. $\delta = W_a^\top K(s)$;

### A. LEARNING-BASED DECISION MAKING USING USP-KLSPI IN COMPLEX TRAFFIC ENVIRONMENTS

#### 1) USP-KLSPI

In order to improve the sampling efficiency and reduce the training time, this article proposes an USP-KLSPI algorithm for decision making.

Uneven sampling is reflected in collecting more samples from the interest intervals, while roughly sampling in the uninterested area. This sampling method is similar to the attention mechanism of human in life. For instance, when pedestrians go across the road, they tend to pay attention to the information of vehicles that are very close. Assuming that the distance ranges between the ego vehicle and social vehicle are:

$$d_f \in [0, d_{f\max}], d_r \in [0, d_{r\max}] \tag{7}$$

where $d_{f\max}$ and $d_{r\max}$ are related to the maximum perception distance of the ego vehicle. The sampling ranges are composed by

$$\begin{cases} D_{fi} = \omega_{fi} \cdot d_{f\max}, & \omega_{fi} \le 1 \\ D_{ri} = \omega_{ri} \cdot d_{r\max}, & \omega_{ri} \le 1 \end{cases} \tag{8}$$

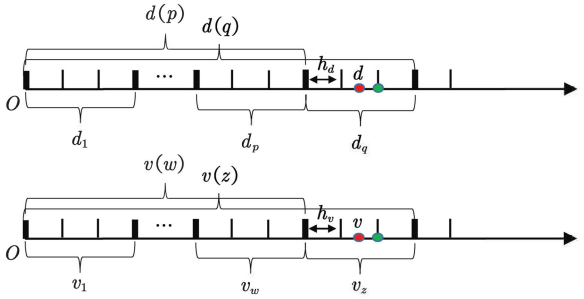



**FIGURE 2.** Sample pooling process: In the upper part of the figure, the red point *d* is a distance element in the sample and the green point is the corresponding pooling result. In the lower part of the figure, the red point *v* is a speed element in the sample and the green point is the corresponding pooling result.

where $\omega_{fi}, \omega_{ri}$ are the important factors and can be adjusted by hand according to the heuristic information. We initialize the distances of social vehicles in $D_{fi}$ and $D_{ri}$ while collecting the $i$-th sub-sample set. A data sample collection process is performed by using an exploration policy, which has the form of $[s_t, a_t, s_{t+1}, r_t]$. Repeat the above processes until the desired number of samples are collected for $N$ sub-sample sets.

To obtain the representative information, we use the pooling strategy to process elements in each sample. The intuitive pooling process is shown in Fig.2. As seen from it, both $d(n)$ and $v(n)$ represent the sum of $n$ intervals. They are described as follows:

$$d(n) = \sum_{i=1}^{n} d_i, v(n) = \sum_{i=1}^{n} v_i \tag{9}$$

where $d_i$ and $v_i$ represent the $i$-th interval size. For states $d \in [d(p), d(q)]$ and $v \in [v(w), v(z)]$, we assume the small intervals in $d_q$ and $v_z$ are with constant values ($h_d$ and $h_v$). Elements $d$ and $v$ in a sample can be processed with a max-pooling manner:

$$d = d(p) + \lceil \frac{d - d(p)}{h_d} + 1 \rceil * h_d \tag{10}$$

$$v = v(w) + \lceil \frac{v - v(w)}{h_v} + 1 \rceil * h_v \tag{11}$$

With the processed samples, we train the KLSPI algorithm for a decision-making policy. According to [37], KLSPI uses the way of kernel feature representation to approximate state-action value function $Q^{\pi}$. To a state $s_i$, a vector of basis functions can be expressed as follows:

$$K(s_i) = \left(\bar{k}(s_i, s_1), \bar{k}(s_i, s_2), \ldots, \bar{k}(s_i, s_c)\right)^{\top} \tag{12}$$

where $\bar{k}(\cdot, \cdot)$ is the kernel function and $K \in \mathbb{R}^{c \times 1}$. Note that, the ALD method could produce kernel dictionary with multiple center points $\mathcal{C} = \{s_1 \cdots, s_c\}$ from the sample set, where $c$ is number of center points. According to the equation (5), the following matrices update rule can be obtained:

$$\begin{aligned} \widehat{\mathbf{A}} &= \widehat{\mathbf{A}} + \sum_{i=1}^{T} K(s_i) \left(K^{\top}(s_i) - \gamma K^{\top}(s_{i+1})\right) \\ \widehat{\mathbf{b}} &= \widehat{\mathbf{b}} + \sum_{i=1}^{T} K(s_i) r_i \end{aligned} \tag{13}$$

$s_k = [s_i, a_i, s_{i+1}, r_i]\,(k = 1, \cdots, T)$, where $s_k$ represents the training samples and $T$ is the number of samples.

With the solved least-squares solution $w^{\pi}$ by (5), the approximation of state-action value function at state $s_t$ can be obtained:

$$\widehat{Q}^{\pi}(s_t) = K^{\top}(s_t) w^{\pi} \tag{14}$$

Hence, the policy that maximizes the state-action value function is the optimal solution:

$$\pi^*(s_t) = \arg\max_{a_t} \widehat{Q}^{\pi}(s_t, a_t) \tag{15}$$

*Remark 1:* It is highlighted that the time cost required for the sampling process of Algorithm 2 is $T \cdot \Delta t$, where $\Delta t$ is the preset time of completing a sampling process and $T = N \cdot N_{num}$.

*Remark 2:* The flops required for the computation in equations (13) and (5) are about $2(T-1)c^2 + (T+1)c$ and $2c^2$, respectively. The flops required for the computation in the

**Algorithm 2** USP-KLSPI for Decision-Making Policy

\\ **Sampling process**
**Require:** The environment simulated with 14-DOF vehicle dynamics, scenario information, and the decision-making task.
**Ensure:** The training samples.
**for** $m \leftarrow 1$ to $N$ **do**
  Determine the sampling intervals by equation (8);
  Randomize each state $s_t$ and collect $N_{num}$ samples $[s_t, a_t, s_{t+1}, r_t]$ by using an exploration policy;
**end for**
\\ **Training process**
**Require:** The training samples, initialized policy weight $w_1^{\pi}$.
**Ensure:** policy weight $w^{\pi}$ for decision making.
\\ $N_{iter}$ is the preset number of iterative learning;
\\ $\epsilon$ is the preset small threshold;
Use sample pooling strategy to process the samples by equations (10) and (11);
**for** $m \leftarrow 2$ to $N_{iter}$ **do**
  Update the weight $w_m^{\pi}$ by KLSPI;
  **if** $\|w_m^{\pi} - w_{m-1}^{\pi}\|_2 \geq \epsilon$ **then**
    Continue.
  **else**
    Let $w^{\pi} = w_m^{\pi}$.
    Output the converged policy weight $w^{\pi}$.
  **end if**
**end for**

off-line training process of Algorithm 2 are about $2Tc^2 + (T+1)c$. The flops count can be reduced approximately as $Tc^2$. This means the algorithm complexity grows linearly with sample numbers $T$, and the computation time is also related to the number of kernel centers $c$.

*Remark 3:* When the trained policy is deployed for real-time decision-making problems, the required flops for computing the optimal policy (i.e. equations (14) and (15)) are $n_a(2c-1)$, where $n_a = 3$ is the number of optional actions. The flops count can be reduced approximately as $c$. The number of kernel centers $c$ in scenario-A is with a magnitude of $10^2$, while is about $10^3$ in scenario-B. Hence it has a low computation time and is feasible to deal with real-time planning problems.

The USP-KLSPI algorithm applied for obtaining the decision-making policy of intelligent vehicles is presented in the Algorithm 2.

### 2) DECISION MAKING IN LEFT-TURN WITHOUT TRAFFIC SIGNALS

With the development of vehicle communication technology, the information on surrounding vehicles can be easily obtained. This article assumes that, within the autonomous vehicle's perception range, we can obtain the speed, position, and current lane information of the social vehicles.

The scenario of left-turn without traffic signals is shown in Fig. 3. The autonomous vehicle needs to cross with consideration of those coming from the front and rear directions. The turning radius of the intersection is an arc with a radius of around $13m$ ($R \approx 13m$). Taking into account the driving safety and the constraints of vehicle dynamics, a reasonable longitudinal speed around $10m/s$ is adopted.

Before sampling, an MDP model needs to be established. As shown in Fig. 3, in the process of driving through each lane, the state in MDP is defined as:

$$s_t = \left[V_{Ego}\ V_{ri}\ d_{ri}\ V_{fi}\ d_{fi}\right] \quad i = 1 \text{ or } i = 2 \tag{16}$$

Among them, $i$ is the lane information, $V_{ri}/V_{fi}$ represent the speed of the nearest rear/front social vehicle; $d_{ri}$ and $d_{fi}$ are the distances from the autonomous vehicle, respectively.

The optional actions of the autonomous vehicle in this scenario can be expressed as:

$$a_t = \{Slow, Keep, Acc\} \tag{17}$$

The actions are slowing down, keeping an original speed and acceleration, respectively. Denote $V_{Ego}$ as the current longitudinal speed of the ego vehicle, where $V_{Ego}$ is set within the range $[0, 13m/s]$. Denoting the notation $L_{Ego}$ as the current lane information of the ego vehicle, a collision is defined as $\left(L_{Ego} == i \&\& d_{fi} \le 5m\right) || \left(L_{Ego} == i \&\& d_{ri} \le 15m\right)$. A successful decision-making process means no collision occurs when $X_{Ego} < -25m \&\& L_{Ego} == 2$.

To bridge the gap between simulated and real environments, social vehicles are considered with 14-DOF vehicle dynamics [40]. They have different desired velocity and a corresponding safe tracking distance $d_{follow}$. The longitudinal

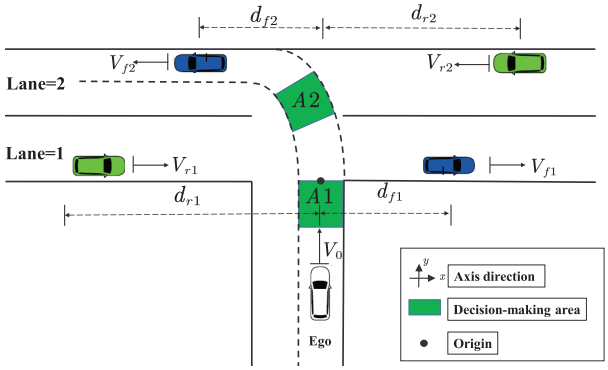



**FIGURE 3.** Left-turn without traffic signals (Scenario-A): When the ego vehicle crosses the 1-th lane, the vehicles on the left are considered as the rear vehicles. When the ego vehicle crosses the 2-th lane, the vehicles on the right are considered as the rear vehicles.

planning policies of the $\tau$-th social vehicle can be expressed as follows:

$$\begin{cases} v_{\tau\_expect} & d_{\tau j} \ge d_{follow} \\ v_{\tau\_brake} & d_{\tau j} < d_{follow} \end{cases} \tag{18}$$

where $v_{\tau\_expect}$ is the expected velocity, $v_{\tau\_brake}$ represents the speed that social vehicle should slow down to. $d_{\tau j}$ is the distance between the two social vehicles. Assuming the $\tau$-th car is following the $j$-th car, $d_{follow}$ can be expressed as follows:

$$d_{follow} = v_\tau \cdot \Delta T + \frac{v_\tau^2}{2a_\tau} - \frac{v_j^2}{2a_j} \tag{19}$$

where $\Delta T$ is the reaction time of the drivers. $v_\tau$ and $v_j$ represent the current speed. $a_\tau$ and $a_j$ are the corresponding brake accelerations. Hence, $d_{follow}$ is a guidance for planning the longitudinal speed of social vehicles.

As shown in Table 1, the detailed parameter settings are given. In the case of multiple forward/backward vehicles in the $i$-th lane, the nearest forward/backward vehicle will be considered into the state $s_t$. When there is no vehicle forward/backward, $d_{fi}$, $V_{fi}$ and $d_{ri}$, $V_{ri}$ are determined by the following equation.

$$\begin{cases} d_{fi} = d_{f\max}, V_{fi} = V_{f\max} & d_{fi} \ge d_{f\max} \\ d_{ri} = d_{r\max}, V_{ri} = V_{r\max} & d_{ri} \ge d_{r\max} \end{cases} \tag{20}$$

Both the surrounding vehicles and the ego vehicle are simulated with 14-DOF vehicle dynamics [40]. After the ego vehicle takes a randomized action $a_t$, one sampling process terminates when the ego vehicle crosses the 1-th lane or waits at the interaction for $1s$, thereby obtaining the next state $s_{t+1}$. The simulation runs for such preset time mainly because we have computed the approximate time to complete a sampling process. The reward function is designed as

$$r_t = \begin{cases} -1000 & \text{collision} = 1 \\ -400 & Y_{Ego} < 0 \&\& \text{collision} = 0 \\ -\xi \cdot \Delta t & \text{collision} = 0 \end{cases} \tag{21}$$

**TABLE 1. Parameter settings of left-turn without traffic signals scenario.**

| Vehicle type | Initial Position | Speed range | Policies | Testing time |
|---|---|---|---|---|
| Ego vehicle | (0,-4) | $V_{Ego} \in [0, 46.8km/h]$ | $\{Slow, Keep, Acc\}$ | $10s$ |
| Front social vehicle | $d_{fi} \in [0, 150m]$ | $V_{fi} \in [0, 70km/h]$ | $\{Slow, Acc\ to\ v_{max}\}$ | 10s |
| Rear social vehicle | $d_{ri} \in [0, 150m]$ | $V_{ri} \in [0, 70km/h]$ | $\{Slow, Acc\ to\ v_{max}\}$ | $10s$ |

**TABLE 2. Parameter settings in multi-lane merging from side-ways scenario.**

| Vehicle type | Initial Position | Speed range | Policies | Testing time |
|---|---|---|---|---|
| Ego vehicle | (0,-2.5) | $V_{Ego} \in [0, 80km/h]$ | $a$ | $10s$ |
| Front social vehicle $(f1)$ | $d_{f1} \in [0, 150]$ | $V_{f1} \in [0, 80km/h]$ | $C_{actions}$ | $10s$ |
| Rear social vehicle $(r1)$ | $d_{r1} \in [0, 100]$ | $V_{r1} \in [0, 80km/h]$ | $C_{actions}$ | $10s$ |
| Front social vehicle $(f2)$ | $d_{f2} \in [0, 150]$ | $V_{f2} \in [0, 90km/h]$ | $A_{actions}$ | $10s$ |
| Rear social vehicle $(r2)$ | $d_{r2} \in [0, 100]$ | $V_{r2} \in [0, 90km/h]$ | $A_{actions}$ | $10s$ |
| Front social vehicle $(f3)$ | $d_{f3} \in [0, 150]$ | $V_{f3} \in [0, 100km/h]$ | $B_{actions}$ | $10s$ |
| Rear social vehicle $(r3)$ | $d_{r3} \in [0, 100]$ | $V_{r3} \in [0, 100km/h]$ | $B_{actions}$ | $10s$ |

where $\xi$ is the adjustability coefficients, i.e. the penalty for the magnitude of the velocity. Variable $\Delta t$ is the completion time of each task, and $Y_{Ego}$ is the ordinate value. The process of obtaining a sample $[s_t, a_t, s_{t+1}, r_t]$: in initial state $s_t$, randomized action $a_t$ was taken, a reward $r_t$ was received, and the resulting state was $s_{t+1}$. Then we train the KLSPI algorithm based on the samples to obtain a decision-making policy.

One thing to emphasize is that the autonomous vehicle needs to make continuous decisions in the area of $A1$ and $A2$ which are shown in Fig. 3.

### 3) DECISION MAKING IN MULTI-LANE MERGING FROM SIDE-WAYS

As shown in Fig. 4, the scenario is composed of three lanes. The autonomous vehicle has to consider the social vehicles' motion in three lanes, especially for the social vehicles in the 2-th lane and the 1-th lane. The autonomous vehicle has to choose a proper occasion to merge into the 1-th lane. In the meanwhile, longitudinal speed is also given by the decision-making layer.

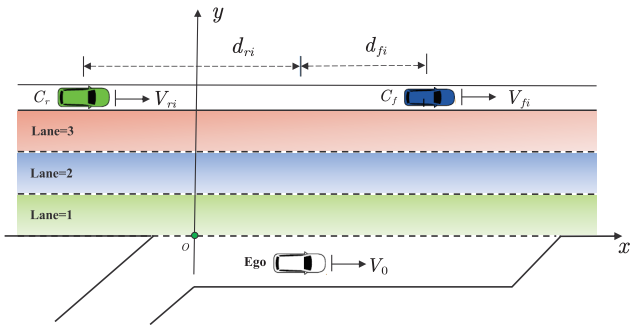



**FIGURE 4. Multi-lane merging from side-ways (Scenario-B).**

As shown in Table 2, the detailed parameter settings are given. In the case of multiple forward/backward vehicles in the $i$-th lane, the nearest forward/backward vehicle will be considered into the state $s_t$. When there is no vehicle forward/backward, $d_{fi}, V_{fi}$ and $d_{ri}, V_{ri}$ are determined by equation (20).

**TABLE 3. Rule-based policies of the *i*-th social vehicle.**

| State | Action | Prerequisite |
|---|---|---|
| $Lane = 1$ | Change to the 2-th lane | $[d_{f1} \leq 40 \cup V_{f1} - V_d \leq -3] \cap d_{f2} - d_{f1} > 10 \cap V_{f2} > V_{f1} \cap d_{r2} > 80$ |
| | Keep the original lane | ELSE |
| $Lane = 2$ | Change to the 3-th lane | $[d_{f3} > 50 \cap V_{f3} \geq V_d]$ |
| | Change to the 1-th lane | $[V_{f1} > V_{f2} \cap d_{r1} > 80]$ |
| | Keep the original lane | ELSE |
| $Lane = 3$ | Change to the 2-th lane | $[d_{f1} > 50 \cap V_{f2} \geq V_d] \cup [V_{f2} > V_{f3} \cap d_{r2} > 80]$ |
| | Keep the original lane | ELSE |

Before sampling, an MDP model needs to be established. The 14-dimensional state $s_t$ is defined as

$$s_t = \left[L_{Ego}\ V_{Ego}\ V_{fi}\ d_{fi}\ V_{ri}\ d_{ri}\right]\ i = 1, 2, 3 \tag{22}$$

The action $a_t$ is defined as

$$a_t \in \{LCT\ (0)\ Acc,\ LCT\ (0)\ Slow,\ LCT\ (1)\} \tag{23}$$

where $LCT\ (i)\ , \{i = 0, 1\}$ means a lane-changing maneuver to the $i$-th lane, $LCT(0)$ represents driving on the original ramp, $LCT(1)$ means a lane-changing maneuver to the 1-th lane. *Acc*, *Slow* mean to accelerate and slow down, respectively. As the equation (23) shows, the autonomous vehicle has a total of three optional actions in this scenario. They are: acceleration on the ramp, slowing down on the ramp, and changing to the 1-th lane. A collision is defined as $\left(L_{Ego} == i \&\& d_{fi} < 5m\right) || \left(L_{Ego} == i \&\& d_{ri} < 20m\right)$.

In Fig. 5, we use the notations $A_{actions}$, $B_{actions}$, $C_{actions}$ to represent the optional actions at the states $A, B, C$, respectively. Under the premise of ensuring driving safety, the driving policies should conform to the driving habits of human drivers as much as possible, and the rule-based policies of social vehicles are presented in Table 3. Through the above settings, the simulation environment gradually becomes high-fidelity.

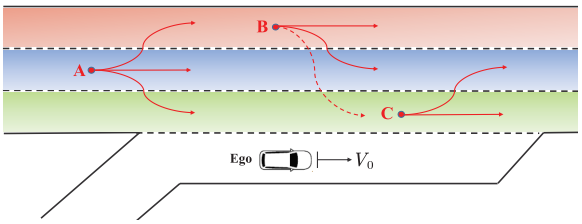



**FIGURE 5.** Decisions of social vehicles in each lane: Solid arrows indicate the optional decisions of social vehicles. The dotted arrow is a sudden maneuver merging into the slow lane. Under these potential collision threats, the ego vehicle needs to make decisions for safe merging.

After the ego vehicle taking a randomized action $a_t$, one sampling process terminates when the ego vehicle merges into the 1-th lane or drives on the ramp for $5s$, thereby obtaining the next state $s_{t+1}$. The reward function is a quantitative evaluation of the behavioral decision. The reward function is designed as:

$$r_t = \begin{cases} -1000 & Collision \\ \max\left(-5 * X_{Ego}, -600\right) & Ego.Lane = 0 \\ -|V_{Ego} - V_d| & Ego.Lane = 1 \end{cases} \quad (24)$$

where $X_{Ego}$ represents the current abscissa of the autonomous vehicle. Due to the limited length of the ramp, a greater penalty will be given when the autonomous vehicle has a larger abscissa value. $V_d$ represents the desired maximum speed of the ego vehicle, and this item is used to force the autonomous vehicle to reach the desired speed faster.

Samples with the form of $[s_t, a_t, s_{t+1}, r_t]$ can be collected and the KLSPI algorithm is trained for a decision-making policy.

### B. THE BATCH-MODE DHP ALGORITHM FOR MOTION PLANNING

At the higher layer, we model the complex dynamic scenarios and propose a USP-KLSPI algorithm to learn the decision-making policy. In dealing with the decision-making problems, the motion data-samples of the ego vehicle and the rule-based surrounding vehicles are collected with a high-fidelity dynamic model for the learning process. Hence, the inconsistency between the decision generated at the higher layer and the operation in the lower planning layer can be avoided. The higher layer has considered the longitudinal speed policy. Therefore, the planning work in the lower layer can be reduced a lot as we only need to plan the trajectory in the lateral direction. Generally speaking, the high-fidelity 14-DOF vehicle model can be employed in the learning process. However, it will bring a computation load. We simplify the high-fidelity 14-DOF vehicle model to the lateral dynamics and utilize DHP algorithm learning in batch-mode manner to realize motion planning. The simplified lateral dynamic model includes the following four-dimensional states $s = [e_{Lat}\ \dot{e}_{Lat}\ \Delta\varphi\ \Delta\dot{\varphi}]^\top$, where $e_{Lat}$ is the lateral error between the reference point and c.g. of the vehicle, and the $\Delta\varphi$ represents the heading error between the vehicle and the expect yaw angle.

**Algorithm 3** Sampling Process of Motion Planning

**Require:** 14-DOF dynamics, $N_{Sam}$, reference $y = \psi(x)$.
**Ensure:** $\{s_1, s_2, \cdots, s_M\}$.

```
for m ← 1 to N_sam do
    State initialization: s = [e_Lat ė_Lat Δφ Δφ̇];
    for n ← 1 to 1000 do
        Randomize u_n in the range of [−u_max, u_max];
        Obtain the next state s_{n+1} with the 14-DOF vehicle
        dynamics;
        if ‖s_{n+1}‖ ∈ s_max then
            Save s_{n+1};
        else
            Break;
        end if
    end for
end for
```

The algorithm of sampling process in motion planning is shown in Algorithm 3, where $N_{Sam}$ represents the desired episodes, and the reference $y = \psi(x)$. As seen from it, the first step is to initialize the state in each state-dimensional range, $s_{max} = [1, 1, \pi/2, \pi/2]$, and the steering angle within the preset range ($u_{max} = \pi/6$) is applied to the 14-DOF dynamics to obtain the next state. In this way, we can obtain the desired samples of simulating the dynamics of the real vehicle.

The learning-based motion planning in this article utilizes a kernel-based DHP [41] to train for a planner. The next state is obtained by applying the control action to the 14-DOF vehicle dynamics in the simulated environment.

The reward function is defined as

$$r = s^\top Q s + u^T R u + p \cdot e^{-\|e_{Lat}\|} \quad (25)$$

The adjustable parameter $p$ is the penalty factor and set as 10. When there are no obstacles, the parameter $p$ is set as 0. Among them, $Q = diag\{1, 1, 5, 1\}$ and $R = 2$. The item $p \cdot e^{-\|e_{Lat}\|}$ indicates that the reward function is smaller with the distance increasing. In practical application, it is not suitable for the ego car to stay too far away from the obstacle (large lateral error $e_{Lat}$ does not meet the requirements of the planning task.). Therefore, we hope to keep a balance. A solution is to set proper penalty matrix $Q$. The item $s^\top Qs$ can limit the lateral error from too large. The other idea is to set the penalty factor $p = 0$ and adopt the reference trajectory changing strategy for realizing obstacle avoidance when it is a structured road.

The batch-mode DHP planner can be trained by data collected from real cars or high-fidelity software Carsim. In fact, collecting training data of decision making through the two ways requires a lot of resources. Therefore, this article utilizes 14-DOF vehicle dynamics for sampling in the simulation environment. After collecting about 30000 samples generated by the 14-DOF dynamic model, we train the Algorithm 4 for a planning policy. When the policy is deployed to real-time

**Algorithm 4** The Training Process for Motion-Planning Policy

**Require:** 14-DOF dynamics, initialized weights of actor and critic modules, i.e. $W_a^{[0]}$, $W_c^{[0]}$.
**Ensure:** motion-planning policy $W_a$.
1: Obtain the desired training samples by Algorithm 3;
2: Set the reward functions by the equation (25);
3: \\ Combine the samples and simplified 14-DOF dynamic model to train the DHP algorithm;
4: \\ $N_{iter}$ is the preset number of iterative learning;
5: **for** $m \leftarrow 1$ to $N_{iter}$ **do**
6: Process the samples in a batch-mode learning manner;
7: Update the weight $W_a^{[m]}$;
8: **end for**
9: Let $W_a = W_a^{N_{iter}}$;

planning tasks, the lateral steering angle can be expressed as

$$u = W_a^{\top} K(s) \tag{26}$$

where $u$ is the front steering angle, i.e. $\delta$, $W_a$ is the weight of the actor module, $K(s)$ can be computed by the format of equation (12). Based on the 14-DOF vehicle dynamics and trained motion-planning policy, we can plan a trajectory of vehicle motion.

*Remark 4:* In this note, we construct the feature by the format of equation (12). Denote the number of samples and kernel centers as $T_m$ and $c_m$, respectively. Through approximation, the algorithm computation complexity in the training process is $T_m c_m^2$.

*Remark 5:* When the offline trained policy is applied to real-time motion planning problem, the lateral planning results can be computed by equation (26). As $W_a \in \mathbb{R}^{c_m \times 1}$ and $K(s) \in \mathbb{R}^{c_m \times 1}$, the computational complexity is about $2c_m - 1$. It can be approximately reduced as $d_m$, where $d_m$ is with a magnitude of 10. Hence it has a low computation complexity and is feasible to deal with real-time planning problems.

## IV. SIMULATION AND ANALYSIS

In this section, the proposed HRLDP algorithm is applied to the left-turn without traffic signals and multi-lane merging from side-ways scenarios. Corresponding simulation results and analyses are described in the following sections. Furthermore, the discussion is also presented.

### A. LEFT-TURN WITHOUT TRAFFIC SIGNALS

The simulation is performed in the Matlab environment by a Desktop with Intel i7-8700K CPU @ 3.7GHz and 16GB RAM, and Windows 10 operating system. We obtain the average time cost of our algorithm in 100 computations, i.e. $t_{de} = 7.32 \times 10^{-4} s$ and $t_{pl} = 2.98 \times 10^{-4} s$, where $t_{de}$ and $t_{pl}$ are the average computation time of our decision-making and motion-planning algorithms, respectively. The simulation results are shown in the Fig.6. We show the positions of the vehicles in a time series, thereby obtaining an $x$-$y$-$t$ diagram. To present the decision-making and motion-planning processes better, Fig.7 shows a screenshot of different moments. As seen from the two figures, the autonomous vehicle waits for the social vehicles at the intersection firstly. After the social vehicle (green diamond in Fig.6 and green cube in Fig.7) just passes, the ego vehicle starts to go across the intersection. Under normal circumstances, the ego vehicle will drive to the 2-th lane safely. In Fig.6 and Fig.7, as the red social vehicle slows down suddenly, the autonomous vehicle then calls for the motion planning module to avoid obstacle. From the trajectories in Fig.6, the autonomous vehicle finally accomplishes the task successfully.

Then this article compares the performance of KLSPI and USP-KLSPI in left-turn without traffic signals scenario. To figure out the influence of parameters $h_d$, $h_v$ on our decision-making task, we determine several representative pooling intervals, i.e. $h_d \in \{1, 10, 20\}$ and $h_v \in \{1, 5, 10\}$ to obtain an approximate optimal combination. Then we perform testing accordingly as seen in Table 4. When both $h_d$ and $h_v$ are 10, the combination has a better performance in training time and obstacle avoidance assistance (OAAR).

**TABLE 4. The influence of parameters $h_d$, $h_v$ on decision-making task.**

| $h_d$ \ $h_v$ | 1 | 5 | 10 |
|---|---|---|---|
| 1 | $(68s, 1.2\%, 5.04s)$ | $(69s, 1.4\%, 5.05s)$ | $(66s, 1.1\%, 5.15s)$ |
| 10 | $(66s, 1.2\%, 5.04s)$ | $(73s, 1.3\%, 5.23s)$ | $(\mathbf{48s}, \mathbf{0.7\%}, \mathbf{5.25s})$ |
| 20 | $(71s, 2.2\%, 5.32s)$ | $(66s, 2.5\%, 5.31s)$ | $(54s, 1.8\%, 5.27s)$ |

$(\cdot, \cdot, \cdot)$ represents training time, OAAR and ATC, respectively.

In USP-KLSPI, the samples are composed of two parts: initial states in 20000 samples are within the intervals as Table 1, i.e. $D_{f1} \in [0, 150]$ and $D_{r1} \in [0, 150]$. Initial states in 10000 samples by uneven sampling are within the relatively small intervals, i.e. $D_{f2} \in [0, 80]$ and $D_{r2} \in [0, 80]$. Hence a total of 30000 samples consisting of two sub-sample sets are collected. In addition, $\xi$ in the reward function is set as 10. Table 5 shows the simulation results and the kernel widths in the different algorithms are the same, i.e. $\sigma = 20$, for a fair comparison. For the performance of OAAR and average time cost (ATC) while completing this task, the two algorithms are similar. Furthermore, the USP-KLSPI performs well in reducing the training time.

**TABLE 5. Performance of two decision-making algorithms.**

| Left-turn without traffic signals (Kernel width:$\sigma = 20$) | | | | |
|---|---|---|---|---|
| Algorithm | Samples | Training time | OAAR[1] | ATC[2] |
| KLSPI | 30000 | 72s | 1.10% | 5.08s |
| USP-KLSPI | 30000 | **48s** | **0.7%** | **5.25s** |

[1] OAAR: Obstacle avoidance assistance rate in 1000 tests.
[2] ATC: the average time cost in a test. If a left-turn task is not completed in $10s$, the time cost is still counted as $10s$.

Under the same test set, we test the USP-KLSPI algorithm and the expert policy (in Table 6). The corresponding results are shown in Table 7. As seen from it, the completion rate

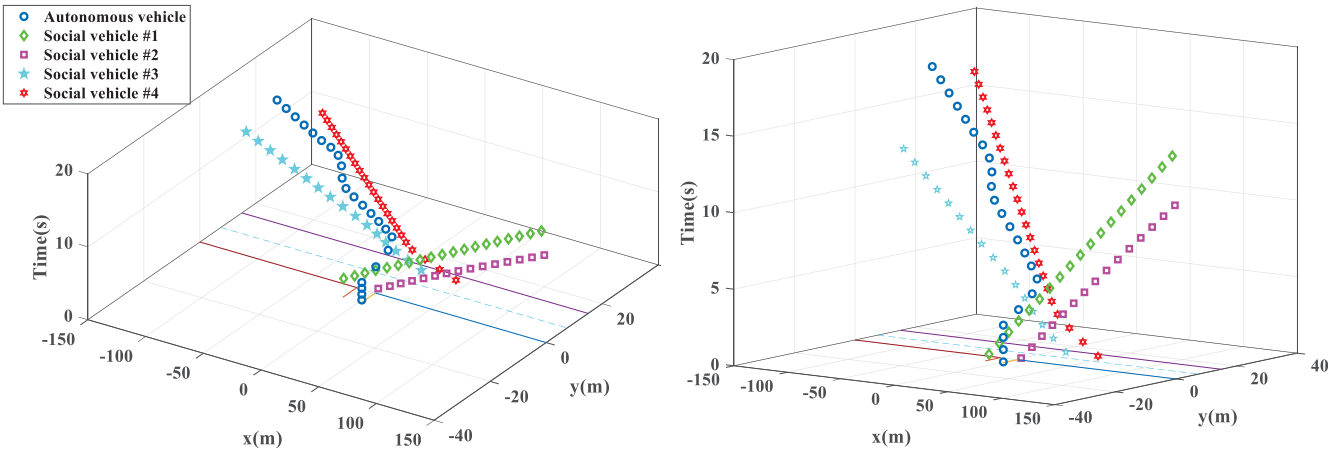



**FIGURE 6.** The *x-y-t* figure of decision making and motion planning in scenario-A: The left and right figures are the two different views of the *x-y-t* figure. The blue circles represent the autonomous vehicle, and the rest shapes are the social (non-autonomous) vehicles.

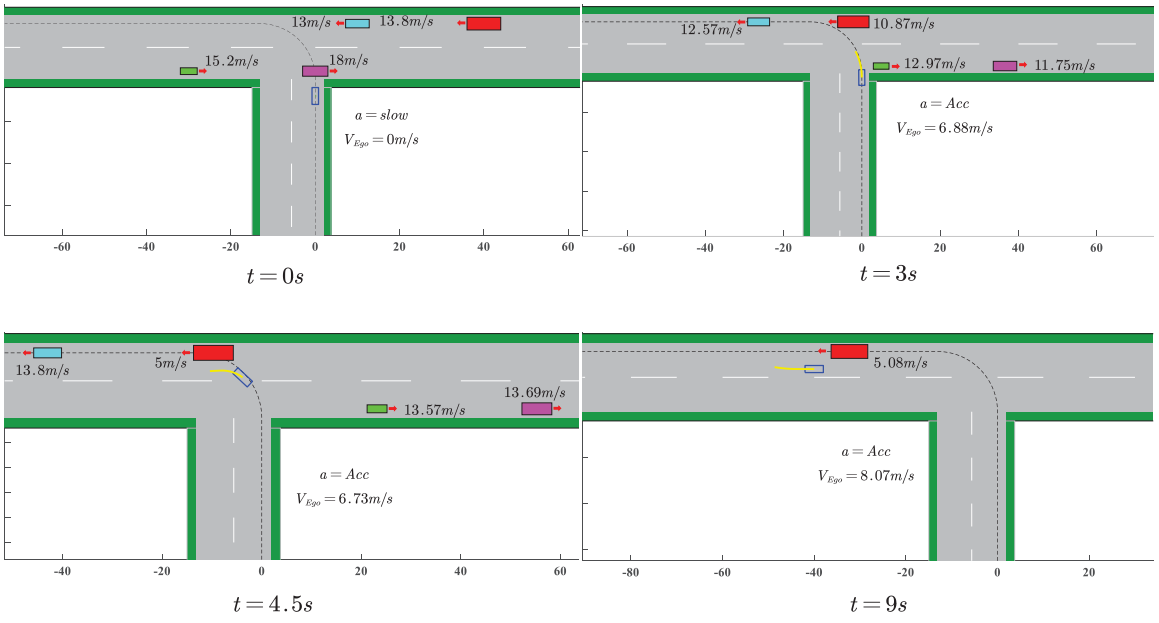



**FIGURE 7.** Decision making and motion planning in scenario-A: The blue transparent cubes are the autonomous vehicle, and the black dotted lines are the reference that the autonomous vehicle will track. The thick yellow lines in front of the autonomous vehicle are the planning results. The rest cubes are the social (non-autonomous) vehicles, and the red arrows are the driving directions.

of expert policy is 82.2%. For USP-KLSPI, about 0.7% of tests need obstacle avoidance assistance and the completion rate is 99.6%. In terms of ATC, the USP-KLSPI has a better performance due to the conservation of the expert policy. That is to say, the autonomous vehicle using our USP-KLSPI algorithm is more flexible in complex dynamic environments.

### B. MULTI-LANE MERGING FROM SIDE-WAYS

In the same line with scenario-A, an extensive simulation is conducted in the multi-lane merging from side-ways scenario. The simulation environment is in the same line with the previous scenario. We obtain the average time cost of our algorithm in 100 computations, i.e. $t_{de} = 4.4 \times 10^{-3}s$ and $t_{pl} = 2.84 \times 10^{-4}s$. The corresponding results are shown in Fig. 8. It shows the decision-making and motion-planning processes with several views. To present the two processes better, we only highlight the vehicles that directly influence the motion of autonomous vehicle, i.e. the red triangle and the green square in Fig.8.

As seen from Fig.8, the blue circles represent the positions of the autonomous vehicle. The initial position of the autonomous vehicle is $(0, -2.5)$. The autonomous vehicle slows down for waiting the green vehicle nearby to pass and prepares to merge into the 1-th lane. Then a red vehicle changes lane from the 2-th lane to the 1-th lane suddenly. The autonomous vehicle cancels the original decision and

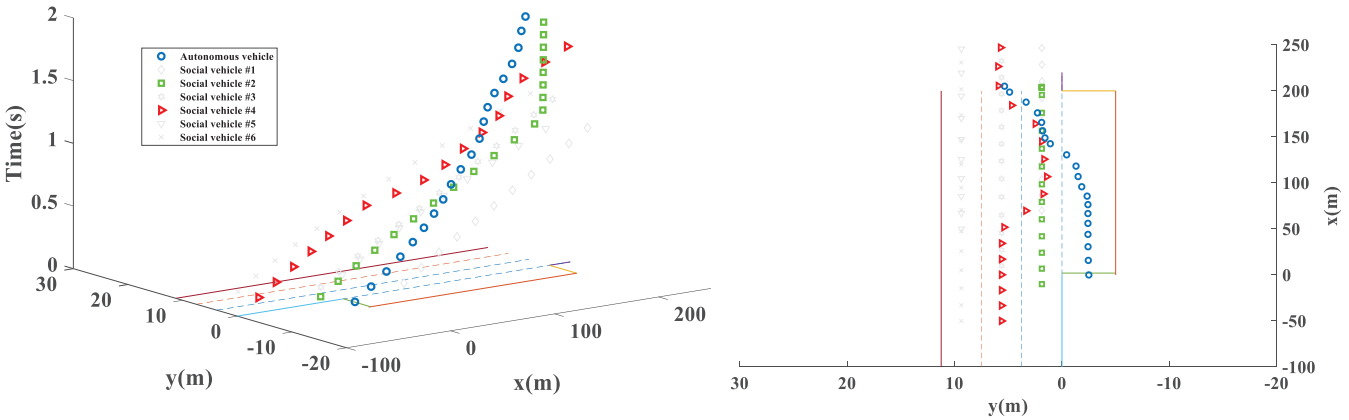



**FIGURE 8.** The *x-y-t* figure of decision making and motion planning in scenario-B: The left and right figures are the two different views of the *x-y-t* figure. The blue circles represent the autonomous vehicle, and the rest shapes are the social (non-autonomous) vehicles. The social vehicles with red triangle and green square that are highlighted for influencing the decision making of the autonomous vehicle.

**TABLE 6.** Expert Policy in scenario of left-turn without traffic signals while crossing the *i*-th lane.

| Action | Prerequisite |
|---|---|
| Stop at the decision-making area | $\frac{d_{ri}}{V_{ri}} < 5s$ |
| Drive through at the desired speed $V_d = 10m/s$ | $5s \leq \frac{d_{ri}}{V_{ri}} \leq 6s$ && $d_{fi} > 30m$ |
| Drive through at the desired speed $V_d = 13m/s$ | $\frac{d_{ri}}{V_{ri}} \geq 6s$ and $d_{fi} > 30m$ |

**TABLE 7.** Comparison of USP-KLSPI and expert policy in left-turn without traffic signals scenario.

| Algorithm | ATC[1] | OAAR | The completion rate[2] |
|---|---|---|---|
| Expert Policy | 7.08s | 30.1% | 82.2% |
| USP-KLSPI | **5.25**s | **0.7**% | **99.6**% |

[1] ATC: the average time cost in 1000 tests. If a left-turn task is not completed in $10s$, the time cost is still counted as $10s$.
[2] The completion rate: If the task is completed within 10 seconds then it is considered a success; Otherwise, it is considered a failure.

**TABLE 8.** The influence of parameters $h_d$, $h_v$ on decision-making task.

| $h_d$ \ $h_v$ | 1 | 5 | 10 |
|---|---|---|---|
| 1 | (**1401s**, **4.9**%, **4.2s**) | (1524$s$, 3.2%, 4.8$s$) | (1748$s$, 2.1%, 5.7$s$) |
| 10 | (2320$s$, 1.5%, 7.1$s$) | (1993$s$, 0.7%, 9.4$s$) | (2318$s$, 0.4%, 13.1$s$) |
| 20 | (3280$s$, 2.4%, 16.5$s$) | (3280$s$, 4.8%, 16.0$s$) | (2813$s$, 6.5%, 15.2$s$) |

$(\cdot, \cdot, \cdot)$ represents training time, OAAR and ATC accordingly.

continues to drive on the original ramp. After the red vehicle passes, the autonomous vehicle merges into the 1-th lane and accomplishes the merging task. To present the processes better, Fig.9 shows a screenshot of different moments and the states information are attached.

Then this article compares the performance of KLSPI and USP-KLSPI in this scenario. To figure out the influence of parameters $h_d$, $h_v$ on our decision-making task, we determine several representative pooling intervals, i.e. $h_d \in \{1, 10, 20\}$ and $h_v \in \{1, 5, 10\}$ to obtain an approximate optimal

**TABLE 9.** Performance of two decision-making algorithms.

| Multi-lane merging from side-ways (Kernel width:$\sigma = 50$) | | | | |
|---|---|---|---|---|
| Algorithm | Samples | Training time | OAAR[1] | ATC[2] |
| KLSPI | 30000 | 1557s | 5.4% | 3.93s |
| USP-KLSPI | 30000 | **1401**s | **4.90**% | **4.23**s |

[1] OAAR: Obstacle avoidance assistance rate in 1000 tests.
[2] ATC: The average time cost in 1000 tests. If a merging task is not completed in $10s$, the time cost is still counted as $10s$.

**TABLE 10.** Expert Policy in scenario of multi-lane merging from side-ways.

| Action | Prerequisite |
|---|---|
| $LCT(0)$ $Slow$ | $\frac{d_{r1}}{V_{r1}} < 3s$ |
| $LCT(1)$ | $\frac{d_{r1}}{V_{r1}} \geq 3s \&\& d_{f1} > 20m$ |

combination. Then we perform testing accordingly as seen in Table 8. When $h_d$ and $h_v$ are set as 1, the combination has a better performance in lower training time and ATC. In USP-KLSPI, the samples are composed of two parts: The initial states of 20000 samples are within the intervals as Table 2, i.e. $D_{f1} \in [0, 150]$ and $D_{r1} \in [0, 100]$. The initial states of 10000 samples by uneven sampling are within the relatively small intervals, i.e. $D_{f2} \in [0, 80]$ and $D_{r2} \in [0, 60]$. Thereby a total of 30000 samples consisting of two sub-sample sets are collected for training. The decision-making policies trained by the two algorithms are employed for testing separately in a total number of 1000. Table 9 shows the simulation results and the kernel widths in different algorithms are the same, i.e. $\sigma = 50$, for a fair comparison. The OAAR and ATC of the two algorithms while completing this task are very close. Furthermore, the proposed USP-KLSPI is useful in reducing the training time. Compared with KLSPI, the OAAR is lower and the ATC is longer. The major reason is that a relatively conservative policy has been learned. While reducing the OAAR, it also sacrifices ATC to complete the tasks.

Because the green vehicle suddenly slows down ahead, the autonomous calls for the motion-planning module to avoid the obstacle. The corresponding results are shown

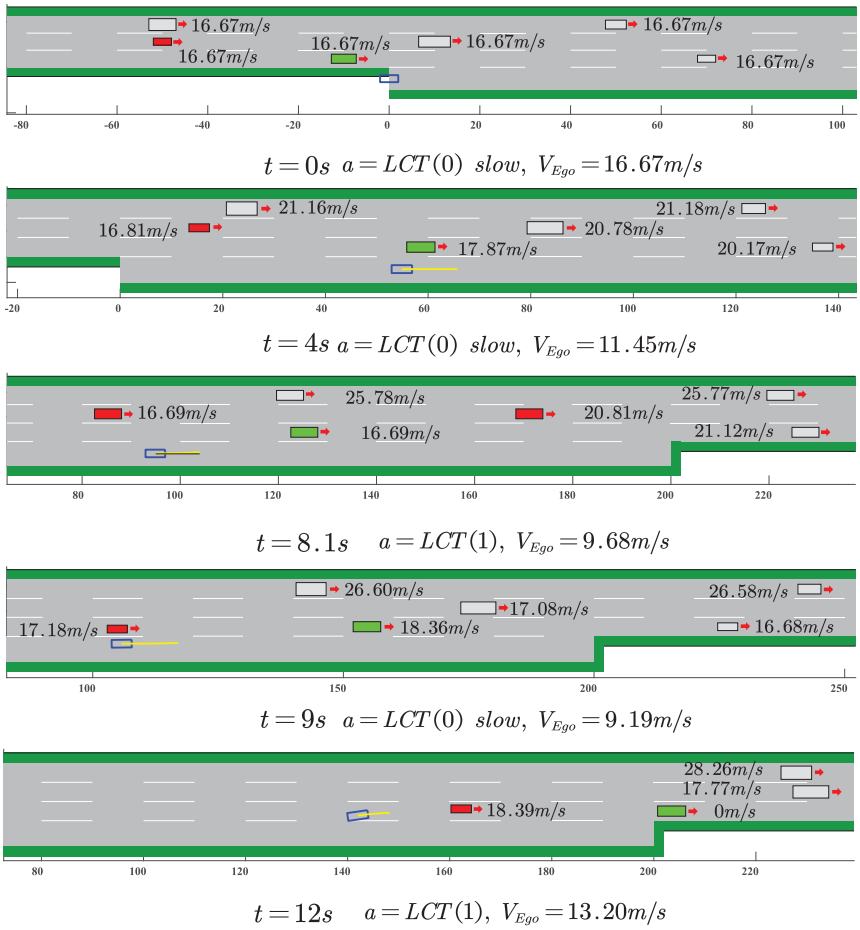



**FIGURE 9.** Decision making and motion planning in scenario-B: The blue transparent cubes are the autonomous vehicle, while the remaining color cubes are the social vehicles. Among them, the red and green cubes are the social vehicles that influence the decision making of autonomous vehicle, and the rest white cubes are the social vehicles that have little influence on autonomous vehicle's decision-making. Correspondingly, the red arrows are the driving directions of the vehicles, and the yellow thick lines are the planed trajectories at current states.

**TABLE 11.** Comparison of USP-KLSPI and expert policy in multi-lane merging from side-ways scenario.

| Algorithm | ATC[1] | OAAR | The completion rate[2] |
|---|---|---|---|
| Expert Policy | 8.41s | 0.2% | 53% |
| USP-KLSPI | **4.23**s | **4.9**% | **88.2**% |

[1] ATC: The average time cost in 1000 tests. If a merging task is not completed in 10$s$, the time cost is still counted as 10$s$.
[2] The completion rate: If the task is completed within 10 seconds then it is considered a success; Otherwise, it is considered a failure.

in Fig.10. As seen from it, a maneuver to the 2-th lane is generated and the autonomous vehicle avoids the obstacle by lateral trajectory planning.

Comparisons with expert policy (as in Table 10) were performed, and the corresponding results are shown in Table 11. The results show the ATC of USP-KLSPI method is much shorter than that of expert policy. In terms of the completion rate, the expert policy has a higher number of failures compared to our proposed method.

### C. DISCUSSION

The algorithms in the two layers are both trained off-line and then deployed to real-time environments. From the analyses of computing performance in remarks 3 and 5, our proposed approach is potential in realizing real-time decision-making and motion planning. To bridge the gap between the simulated and real environment, the data samples used for training are collected using a high-fidelity 14-DOF dynamics in the both layers, and the consistency can be guaranteed. In summary, our algorithm is feasible in dealing with decision-making and motion planning problems in real environments.

As to the applications in real environments, we need to obtain the state information, i.e. $s_t$ as in equation (16) or (23). Then we use the trained policy $w^\pi$ to make real-time decisions by equations (14) and (15). In the aspect of motion planning, as the decisions have included the longitudinal speed information, we only concern about the lateral trajectory planning by equation (26), i.e. $W_a^\top K(s)$. The real-time

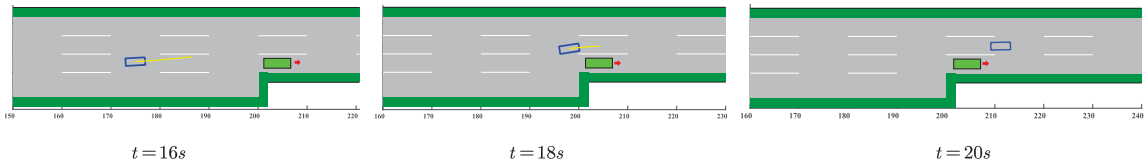



**FIGURE 10.** Motion planning in scenario-B: The blue transparent cube is the autonomous vehicle, and the green cube is the one of the social vehicles. Correspondingly, the red arrows are the direction of the social vehicle and the yellow thick lines are the planned trajectories.

decision-making and motion-planning processes are depicted in the lower part of Figure.1 and in Algorithm 1.

## V. CONCLUSION

In this article, a hierarchical reinforcement learning approach is proposed for autonomous decision making and motion planning in complex dynamic traffic scenarios. The motion data-samples of the ego vehicle and the rule-based surrounding vehicles are collected by a high-fidelity 14-DOF dynamics for the learning process in the decision-making problems. In addition, the decision-making layer considers the optimization of longitudinal speeds. The motion capabilities of the ego vehicle and the surrounding vehicles are evaluated with a high-fidelity dynamic model in the decision-making layer. Therefore, the consistency between the decisions generated at the higher layer and the operations in the lower planning layer can be well guaranteed.

The simulation results on two complex traffic scenarios show the effectiveness and efficiency of our proposed hierarchical reinforcement learning approach. Moreover, comparisons with KLSPI was conducted, and the test results indicate that the proposed USP-KLSPI can reduce the training time obviously. The corresponding results indicate that the autonomous vehicle using the USP-KLSPI algorithm is more flexible in complex dynamic environments. Future work will be focused on efficient feature learning, and we will pay more attention to handling integrated decision and planning problems under complex constraints.

## REFERENCES

[1] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58443–58469, 2020.

[2] G. Xiong, Z. Kang, H. Li, W. Song, Y. Jin, and J. Gong, "Decision–making of lane change behavior based on RCS for automated vehicles in the real environment," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2018, pp. 1400–1405.

[3] I. Batkovic, M. Zanon, M. Ali, and P. Falcone, "Real-time constrained trajectory planning and vehicle control for proactive autonomous driving with road users," in *Proc. 18th Eur. Control Conf. (ECC)*, Jun. 2019, pp. 256–262.

[4] A. Artuãedo, J. Villagra, and J. Godoy, "Real-time motion planning approach for automated driving in urban environments," *IEEE Access*, vol. 7, pp. 180039–180053, 2019.

[5] M. Montemerlo, "Junior: The stanford entry in the urban challenge," *J. field Robot.*, vol. 25, no. 9, pp. 569–597, 2008.

[6] A. Kurt and Ü. Özgäner, "Hierarchical finite state machines for autonomous mobile systems," *Control Eng. Pract.*, vol. 21, no. 2, pp. 184–194, Feb. 2013.

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

[8] X. Xu, L. Zuo, X. Li, L. Qian, J. Ren, and Z. Sun, "A reinforcement learning approach to autonomous decision making of intelligent vehicles on highways," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 50, no. 10, pp. 3884–3897, Dec. 2019.

[9] J. Liu, L. Zuo, X. Xu, X. Zhang, J. Ren, Q. Fang, and X. Liu, "Efficient batch-mode reinforcement learning using extreme learning machines," *IEEE Trans. Syst., Man, Cybern. Syst.*, early access, Aug. 14, 2019, doi: 10.1109/TSMC.2019.2926806.

[10] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 1989, pp. 305–313.

[11] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. L. Cun, "Off-road obstacle avoidance through end-to-end learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2006, pp. 739–746.

[12] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," 2016, *arXiv:1604.07316*. [Online]. Available: http://arxiv.org/abs/1604.07316

[13] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-End learning of driving models from large-scale video datasets," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2174–2182.

[14] Z. Bai, B. Cai, W. ShangGuan, and L. Chai, "Deep learning based motion planning for autonomous vehicle using spatiotemporal LSTM network," in *Proc. Chin. Autom. Congr. (CAC)*, Nov. 2018, pp. 1610–1614.

[15] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 1135–1145, Apr. 2016.

[16] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959.

[17] N. J. Nilsson, "A mobile automaton: An application of artificial intelligence techniques," in *Proc. 1st Int. Joint Conf. Artif. Intell. (IJCAI)*, 1969, pp. 509–520.

[18] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 4, May 1994, pp. 3310–3317.

[19] D. Ferguson and A. Stentz, "Using interpolation to improve path planning: The field D* algorithm," *J. Field Robot.*, vol. 23, no. 2, pp. 79–101, 2006.

[20] A. Nash, K. Daniel, S. Koenig, and A. Felner, "Theta: Any-angle path planning on grids," in *Proc. AAAI*, vol. 7, 2007, pp. 1177–1183.

[21] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime search in dynamic graphs," *Artif. Intell.*, vol. 172, no. 14, pp. 1613–1643, Sep. 2008.

[22] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Dec. 1996.

[23] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, May 2001.

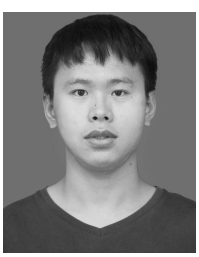
[24] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *Proc. 49th IEEE Conf. Decision Control (CDC)*, Dec. 2010, pp. 7681–7687.

[25] J. Reeds and L. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific J. Math.*, vol. 145, no. 2, pp. 367–393, Oct. 1990.

[26] T. Fraichard and A. Scheuer, "From reeds and Shepp's to continuous-curvature paths," *IEEE Trans. Robot.*, vol. 20, no. 6, pp. 1025–1035, Dec. 2004.

[27] P. Petrov and F. Nashashibi, "Modeling and nonlinear adaptive control for autonomous vehicle overtaking," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 4, pp. 1643–1656, Aug. 2014.

[28] J. P. Rastelli, R. Lattarulo, and F. Nashashibi, "Dynamic trajectory generation using continuous-curvature algorithms for door to door assistance vehicles," in *Proc. IEEE Intell. Vehicles Symp. Proc.*, Jun. 2014, pp. 510–515.
[29] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *Int. J. Robot. Res.*, vol. 29, no. 5, pp. 485–501, Apr. 2010.
[30] J. Ziegler, "Making bertha drive—An autonomous journey on a historic route," *IEEE Intell. Transp. Syst. Mag.*, vol. 6, no. 2, pp. 8–20, May 2014.
[31] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of vol. 1, p. 43scale rc cars," *Optim. Control Appl. Methods*, vol. 36, no. 5, pp. 628–647, 2015.
[32] C. Lian and X. Xu, "Motion planning of wheeled mobile robots based on heuristic dynamic programming for WCICA 2014 proceedings published by IEEE," in *Proc. 11th World Congr. Intell. Control Autom.*, Jun. 2014, pp. 576–580.
[33] S. Al Dabooni and D. Wunsch, "Heuristic dynamic programming for mobile robot path planning based on dyna approach," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2016, pp. 3723–3730.
[34] L. Qian, X. Xu, Y. Zeng, and J. Huang, "Deep, consistent behavioral decision making with planning features for autonomous vehicles," *Electronics*, vol. 8, no. 12, p. 1492, Dec. 2019.
[35] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *J. Mach. Learn. Res.*, vol. 4, pp. 1107–1149, Dec. 2003.
[36] J. A. Boyan, "Technical update: Least-squares temporal difference learning," *Mach. Learn.*, vol. 49, nos. 2–3, pp. 233–246, 2002.
[37] X. Xu, D. Hu, and X. Lu, "Kernel-based least squares policy iteration for reinforcement learning," *IEEE Trans. Neural Netw.*, vol. 18, no. 4, pp. 973–992, Jul. 2007.
[38] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3675–3683.
[39] W. Ding, S. Li, H. Qian, and Y. Chen, "Hierarchical reinforcement learning framework towards multi-agent navigation," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2018, pp. 237–242.
[40] R. Zheng, C. Liu, and Q. Guo, "A decision-making method for autonomous vehicles based on simulation and reinforcement learning," in *Proc. Int. Conf. Mach. Learn. Cybern.*, Jul. 2013, pp. 362–369.
[41] J. Liu, Z. Huang, X. Xu, X. Zhang, S. Sun, and D. Li, "Multi-kernel online reinforcement learning for path tracking control of intelligent vehicles," *IEEE Trans. Syst., Man, Cybern. Syst.*, early access, Feb. 7, 2020, doi: 10.1109/TSMC.2020.2966631.



**YANG LU** received the B.S. degree in automation from the School of Mechanical, Electrical and Information Engineering, Shandong University, Weihai, China, in 2018. He is currently pursuing the M.S. degree in control science and engineering with the College of Intelligence Science and Technology, National University of Defense Technology, Changsha, China. His research interests include reinforcement learning, model predictive control, trajectory planning, behavior decision, and autonomous vehicles.

**XIN XU** (Senior Member, IEEE) received the B.S. degree in electrical engineering from the Department of Automatic Control, National University of Defense Technology (NUDT), Changsha, China, in 1996, and the Ph.D. degree in control science and engineering from the College of Mechatronics and Automation, NUDT, in 2002.

He has been a Visiting Professor with The Hong Kong Polytechnic University, the University of Alberta, the University of Guelph, and the University of Strathclyde, U.K. He is currently a Professor with the College of Intelligence Science and Technology, NUDT. He has coauthored more than 160 papers in international journals and conferences and coauthored four books. His research interests include intelligent control, reinforcement learning, approximate dynamic programming, machine learning, robotics, and autonomous vehicles.

Dr. Xu is a member of the IEEE CIS Technical Committee on Approximate Dynamic Programming and Reinforcement Learning (ADPRL) and the IEEE RAS Technical Committee on Robot Learning. He received the Fork Ying Tong Youth Teacher Fund of China, in 2008, and the Second Class National Natural Science Award of China, in 2012. He serves as the Co-Editor-in-Chief for *Journal of Intelligent Learning Systems and Applications* and the Associate Editor-in-Chief for *CAAI Transactions on Intelligence Technology* (Elsevier). He is an Associate Editor of *Information Sciences*, *Intelligent Automation and Soft Computing*, and *Acta Automatica Sinica*. He was a Guest Editor of the *International Journal of Adaptive Control and Signal Processing* and *Mathematical Problems in Engineering*.

**XINGLONG ZHANG** (Member, IEEE) was born in Anhui, China, in 1990. He received the Ph.D. degree in system and control from the Politecnico di Milano, in 2018. He is currently an Assistant Professor with the National University of Defense Technology, China. His research interests include adaptive dynamic programming, model predictive control, and learning-based control.

**LILIN QIAN** received the B.S. degree in electrical engineering and automation from PLA Air Force Aviation University, China, in 2015, and the Ph.D. degree in control science and engineering from the College of Intelligence Science and Technology, NUDT, in 2020. He is currently a Lecturer with the Unmanned System Technology Research Center, NIIDT. His research interests include reinforcement learning, trajectory planning, behavior decision, and autonomous vehicles.

**XING ZHOU** received the B.Eng. degree in computer science from Hunan University, Changsha, China, in 2012, and the M.S./Ph.D. degrees in computer science/software engineering from the College of Computer, National University of Defense Technology (NUDT), in 2014 and 2019, respectively. He is currently an Assistant Professor with NUDT. He has published more than ten articles in recent years. His research interests include robotics, optimization, and artificial intelligence.