A Reinforcement Learning Approach to Autonomous Decision Making of Intelligent Vehicles on Highways

Xin Xu, *Senior Member, IEEE*, Lei Zuo, Xin Li, Lilin Qian, Junkai Ren, and Zhenping Sun

***Abstract*—Autonomous decision making is a critical and difficult task for intelligent vehicles in dynamic transportation environments. In this paper, a reinforcement learning approach with value function approximation and feature learning is proposed for autonomous decision making of intelligent vehicles on highways. In the proposed approach, the sequential decision making problem for lane changing and overtaking is modeled as a Markov decision process with multiple goals, including safety, speediness, smoothness, etc. In order to learn optimized policies for autonomous decision-making, a multiobjective approximate policy iteration (MO-API) algorithm is presented. The features for value function approximation are learned in a data-driven way, where sparse kernel-based features or manifold-based features can be constructed based on data samples. Compared with previous RL algorithms such as multiobjective Q-learning, the MO-API approach uses data-driven feature representation for value and policy approximation so that better learning efficiency can be achieved. A highway simulation environment using a 14 degree-of-freedom vehicle dynamics model was established to generate training data and test the performance of different decision-making methods for intelligent vehicles on highways. The results illustrate the advantages of the proposed MO-API method under different traffic conditions. Furthermore, we also tested the learned decision policy on a real autonomous vehicle to implement overtaking decision and control under normal traffic on highways. The experimental results also demonstrate the effectiveness of the proposed method.**

***Index Terms*—Autonomous decision-making, intelligent driving vehicles, Markov decision processes (MDPs), multiobjective, reinforcement learning (RL), value function approximation.**

I. Introduction

OVER the past decade, traffic safety and environmental pollution caused by car consumption have become a serious problem in the global world. As a result, the development of intelligent vehicles has been considered as an important solution to the above problem and drawn world-wide interests [1]–[6]. The objective of intelligent vehicles is to use intelligent sensing, decision and control techniques in driving tasks to make road transportation safer, more efficient, and more sustainable [7]. An intelligent vehicle system consists of several functional modules, such as sensing and map-building, decision making, path planning, and motion control. In particular, the autonomous decision making module is one of the most critical modules, which can enable intelligent vehicles to choose appropriate driving maneuvers, like changing lanes, overtaking other vehicles [8]–[10], etc. Due to the complexity and uncertainty of real-world traffic, to realize autonomous decision making of intelligent vehicles is still very challenging.

Until now, various efforts have been devoted to address the decision-making problem for intelligent vehicles. In general, previous decision making approaches for intelligent vehicles can be divided into three classes. The first class is rule-based decision making based on expert systems or fuzzy logic [12]–[14]. Niehaus and Stengel [1] proposed a rule-based decision system for intelligent vehicles on freeways. In this system, the worst-case decision making method was used to deal with the uncertainties in decision-making. In [12] and [13], a prediction and cost function-based algorithm (PCB) was proposed to achieve highway driving for autonomous vehicles. In the PCB algorithm, a prediction engine was built to estimate the intentions of surrounding vehicles and a cost function library was used to find an appropriate driving behavior. Pérez *et al.* [14] proposed a decision approach using fuzzy logic for autonomous overtaking behaviors. Although rule-based decision methods are easy to implement, they usually require lots of prior knowledge and are not robust and adaptive to different traffic situations.

The second class of decision making methods make use of probabilistic models to deal with uncertainties [4], [7], [15]. Ulbrich and Maurer [9] proposed a probabilistic online lane-change decision framework for automatic driving on highways, which can model uncertainties in the lane change decision making process. Schubert [7] and Schubert and Wanielik [15] proposed a decision making approach based on Bayesian networks to consider the uncertainty from perception to the decision stage. However, it is difficult for Bayesian network models to be adaptive to complex hybrid models and dynamic decision tasks.

The third class of decision-making approaches employs various machine learning algorithms to establish decision functions based on observation data. Although supervised learning can be used to learn decision policies by using

Manuscript received October 17, 2017; revised March 23, 2018; accepted August 18, 2018. This work was supported by the National Natural Science Foundation of China under Grant 61751311, Grant U1564214, and Grant 61611540348. This paper was recommended by Associate Editor Z. Liu. *(Corresponding author: Xin Xu.)*

The authors are with the Institute of Unmanned Systems, College of Intelligence Science and Technology, National University of Defense Technology, Changsha 410073, China (e-mail: xinxu@nudt.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2018.2870983

human experiences, it is difficult to collect enough labeled data from skilled drivers. As an important class of machine learning methods, reinforcement learning (RL) [16], [17] is a framework of solving sequential decision making problems in a self-learning style. One advantage of RL methods is that they can learn optimized policies without much model information. In addition, an RL agent can learn an optimal or near-optimal policy by interacting with the environment. In order to design overtaking policies, a multigoal decision method based on tabular RL called Q-learning (QL) was proposed in [18] and [19]. However, the learning efficiency and generalization ability of QL have been shown to be low and function approximation in RL needs to be studied. Zheng *et al.* [20] developed a decision making method based on least-squares policy iteration (LSPI) [21] for autonomous driving but only single decision objective and simple traffic scenarios were considered. In addition, previous works on RL-based decision making [20], [41] have not been tested in real autonomous vehicles.

In this paper, we present a novel RL approach with value function approximation and feature learning for autonomous decision making of intelligent vehicles on highways. First, we model the driving decision making problem as a Markov decision process (MDP). Different performance measures in driving decisions are considered, including safety, smoothness, and speediness. Two different kinds of reward functions are designed in the MDP. One combined reward function is designed for single objective RL methods and the other considers different objectives separately. To learn optimized policies, a multiobjective approximate policy iteration (MO-API) algorithm is presented. The features for value function approximation are learned in a data-driven way, where sparse kernel-based features or manifold-based features can be constructed based on data samples.

Compared with previous RL methods such as multiobjective QL, the proposed MO-API algorithm uses data-driven feature representation for value and policy approximation so that better learning efficiency can be achieved. A highway simulation environment using a 14 degree-of-freedom (DOF) vehicle dynamics model was established to generate training data and test the performance of different decision-making methods for intelligent vehicles on highways. The results illustrate the advantages of the proposed MO-API method under different traffic conditions. Furthermore, we also tested the learned decision policy on a real autonomous vehicle to implement overtaking decision and control under normal traffic on highways. The real-time experimental results also demonstrate the effectiveness of the proposed method.

The main contributions of this paper can be summarized as follows.

1) We present an MO-API algorithm, where the features for value function approximation are learned in a data-driven way. Instead of using tabular representations or hand-crafted features, the proposed approach can use sparse kernel-based features or manifold-based features that are learned from data samples.
2) We established a highway simulation environment using a 14 DOF vehicle dynamics model to generate training data samples offline and test the performance of different decision-making methods for intelligent vehicles on highways. The results illustrate the advantages of the proposed MO-API method under different traffic conditions.
3) The learned decision policy was tested on an intelligent driving vehicle to implement real-time overtaking decision under real traffic on highways. The experimental results also demonstrate the effectiveness of the proposed method.

The rest of this paper is organized as follows. Section II provides some background information about the MDP, RL, and approximate policy iteration (API). Section III presents the details of the MO-API approach for solving the autonomous decision-making problem of intelligent vehicles, including the description of the overtaking problem, the formulation of the MDP model, and the MO-API algorithm with feature learning. In Section IV, the simulation environment is introduced and the simulation results are given. Moreover, real-time decision-making experiments on the HQ-3 autonomous vehicle were also conducted. Finally, Section V draws the conclusion and suggests the future work.

II. Background and Related Work

A. Markov Decision Process

The MDP is a fundamental formulation for RL problems [16], [21], [23]. An MDP is defined by a tuple $\{X, A, P, R\}$, where X is the state space, A is the action space, and R is the reward function. P is the transition model of the MDP. A policy π is a function mapping from the state space to the action space $\pi : X \rightarrow A$, which is used to select actions for every state. The performance of a given policy π in a state x is defined in terms of the expected future rewards. The expected total reward is called the state-value function, which is defined as

$$V^{\pi}(x) = E_{\pi}\left[\sum_{i=0}^{\infty} \gamma^{i} r_i | x_0 = x\right] \tag{1}$$

where γ is the discount factor.

There is another definition of value functions for a given policy π, which is called the action-value function. The action-value function is defined as the expected total reward when taking action a in state x and following the policy π thereafter:

$$Q^{\pi}(x, a) = E_{\pi}\left[\sum_{i=0}^{\infty} \gamma^{i} r_i | x_0 = x, a_0 = a\right]. \tag{2}$$

Since the value functions indicate how good the policy is, they can be employed to find the optimal actions in MDPs. In such cases, the optimal value function is computed at first, and then an optimal action can be determined by a greedy strategy accordingly

$$\pi^{*}(x) = \arg\max_{a} Q^{*}(x, a). \tag{3}$$

B. Multiobjective Reinforcement Learning

Multiobjective optimization (MOO) problems are very popular in a variety of fields such as industries, economics, and

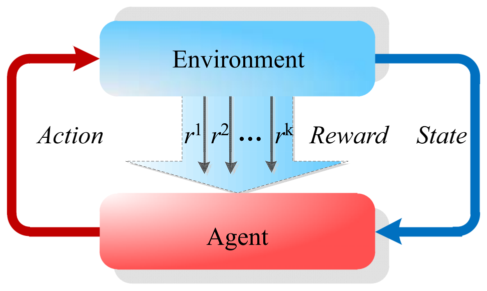


Fig. 1. Diagram of MORL.

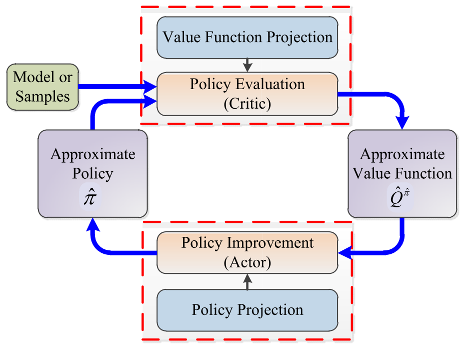

Value Function Projection
Model or Samples
Policy Evaluation (Critic)
Approximate Policy $\hat{\pi}$
Approximate Value Function $\hat{Q}^{\hat{\pi}}$
Policy Improvement (Actor)
Policy Projection

Fig. 2. Flowchart of approximate policy iteration [21].

so on [24]–[27]. Since the objectives in MOO are usually conflicting, it is hard to simultaneously optimize each objective. Thus, it is necessary to find an appropriate tradeoff between the conflicting objectives [28].

Similar to MOO problems, multiobjective RL (MORL) tries to solve sequential decision making problems with multiple objectives by interacting with the environment [28], [29]. In traditional RL, there is only one objective function that needs to be optimized, which is computed based on one single reward from the environment.

In MORL, the learning agent receives more than one reward from the environment at each step according to different objectives, as shown in Fig. 1. Assume k rewards r_i $(i = 1, 2, \dots, k)$ are received by the agent at each step, then there are k different corresponding objectives. Given a certain policy π, there is one action-value function like (2) for each objective. In general, some of the objectives are conflicting. Thus, MORL and MOO has some common issues, like the preferences of the objectives, the representation of preferences, and the approximation of the Pareto front. The task of MORL is to solve the sequential decision-making problems with multiple objectives by learning from the experiences or samples.

C. Approximate Policy Iteration

Policy iteration is an iterative procedure of discovering the optimal policy for a given MDP with known model information [30]. However, in many practical problems, the state transition model and the reward function of MDPs are usually unknown. In such cases, API has been studied, which is based on the samples from the actual process or a generative model of the process [17], [21], [22], [31], [32].

In RL, API algorithms aim at finding the optimal or near-optimal policy using samples generated during the interaction between the learning system and the environment. API mainly consists of two interactive parts. One is called policy evaluation, known as the critic, computing the action-value function of the current policy using temporal difference learning which usually employs projection-based methods for batch-mode learning. The other is called policy improvement, also known as the actor, evaluating all actions for every state to discover possible improvements of the current policy. Fig. 2 shows a diagram of API and the relations between the different parts.

There are two classes of API methods with different value function approximation architectures: linear and nonlinear. The LSPI algorithm is one popular API algorithm using linear architectures [21], which uses the least-squares temporal difference algorithm [33] for policy evaluation. The basis function construction is a key issue in LSPI, which will greatly influence its performance. The kernel LSPI method uses sparse kernel-based features to improve the performance of API [31]. Mahadevan and Maggioni [32] proposed the manifold-based features for value function approximation and designed an improved LSPI approach called representation policy iteration (RPI). There are also other efforts devoted to the improvements of API methods [17], [34], [35]. In this paper, we will design and test multiobjective API methods to solve the autonomous decision-making problem for intelligent vehicles.

III. MO-API Approach for Decision-Making

Due to the complexity and uncertainties of vehicle dynamics and dynamic traffic, it is very difficult for intelligent vehicles to find optimal driving decisions. In this paper, we use RL approaches with VFA and feature learning to deal with this challenge. The RL approach can be model-free and learn from simulated observation samples which are obtained during the interaction with the environment. Moreover, the proposed RL approach has the capability of dealing with uncertainties and complexities of the environment.

A. Decision-Making Task

An intelligent vehicle needs to complete different driving tasks like lane following, lane changing, and overtaking in dynamic, complex environments [19]. For this purpose, the intelligent vehicles must have the ability of making appropriate driving decisions in different traffic situations. Particularly, deciding when to perform the overtaking behavior is very challenging because of the uncertainties and complexities in the traffic. Fig. 3 shows the process of a typical overtaking behavior in two-lane environments. In general, the overtaking process usually includes three stages.

1) Changing lane to the left [Fig. 3(a)].
2) Driving on the left lane (fast speed lane) and passing the vehicle in the adjacent lane [Fig. 3(b)].
3) Changing back from the left lane to the right lane [Fig. 3(c)].

In the overtaking problem, we focus on whether the vehicle can make correct overtaking decisions and in what way we

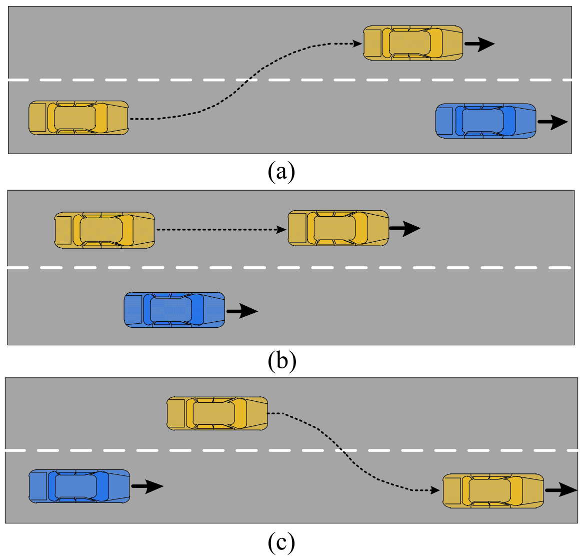


Fig. 3. Three stages in a typical overtaking behavior, the yellow one is the intelligent vehicle and the blue is the slower vehicle. (a) Changing to the left lane. (b) Overtaking the slower vehicle. (c) Changing back to the right lane.

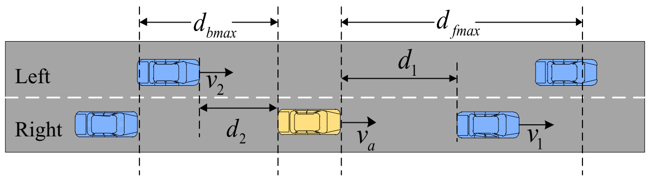


Fig. 4. Two-lane environment. The yellow one is the intelligent vehicle.

evaluate the overtaking process. Three objectives have been considered, i.e., safety, speediness, and smoothness. Based on the three objectives, the driving decision-making process is modeled as a multiobjective MDP (MOMDP).

Note that it is necessary to consider the priority of different objectives according to different preferences. For example, in some cases, the speediness is prior to the smoothness. In some situations, different objectives may be conflicting. However, the safety objective must take the first priority over other objectives. Based on the MOMDP, the MO-API approach will be developed to solve the overtaking decision problem.

As shown in Fig. 4, it is assumed that one direction of the highway consists of two lanes, i.e., the left lane for overtaking and the right lane for normal driving. The states of the environmental vehicles around the intelligent vehicle can be measured. As shown in Fig. 4, $d_{f\max t}$ and $d_{b\max}$ stand for the maximum perception distances of the front sensors and the backward sensors, respectively. The vehicles which fall in this range will be sensed by the intelligent vehicle. d_1 denotes the distance between the intelligent vehicle and the nearest vehicle in the forward direction, and v_1 is the velocity of the front vehicle. d_2 denotes the distance between the intelligent vehicle and the nearest backward vehicle.

B. Dynamics Model of the HQ-3 Autonomous Vehicle

In this paper, the dynamics of an autonomous HQ-3 vehicle platform will be established before the simulation and policy learning process. The HQ-3 vehicle was equipped with a customized autonomous driving system. The vehicle dynamics model has 14 DOF and is constructed based on the real data of the vehicle [20].

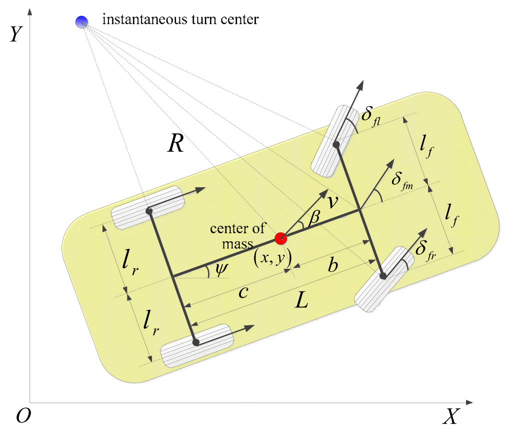


Fig. 5. Schematic of the vehicle kinematics.

Fig. 5 shows the kinematics model of the vehicle. When the vehicle is running around the instantaneous turn center, the relationships between the attitude angle of the vehicle (δ_{fm}) and the steering angles of the front wheels (δ_s) can be defined as follows:

$$\delta_s = \alpha \cdot \delta_{fm} \tag{4}$$

$$\begin{cases} \delta_{fl} = \arctan \frac{L \tan \delta_{fm}}{L - l_f \tan \delta_{fm}} \\ \delta_{fr} = \arctan \frac{L \tan \delta_{fm}}{L + l_f \tan \delta_{fm}} \end{cases} \tag{5}$$

where δ_{fr} denotes the equivalent angle of the right front wheel and δ_{fl} is the left one. l_f stands for the vertical distance from the front wheels to the center of mass and L is the distance between the front axle and the rear axle. α is a constant.

The rolling motion model of the wheels is

$$I_w \dot{\omega} = T_t - T_b - f_x R_l \tag{6}$$

where I_w is the inertia moment of the wheel, T_t is the driving torque, T_b is the braking torque, f_x is the longitudinal force of the wheel, and R_l is the load radius. Then, the input on the steering wheel can be mapped to the equivalent angle of each front wheel. Moreover, the overall force in the dynamics model can be obtained by

$$\boldsymbol{F} = \boldsymbol{T} \begin{bmatrix} 0 \\ 0 \\ \delta_{fr} \end{bmatrix} F_{fl} + \boldsymbol{T} \begin{bmatrix} 0 \\ 0 \\ \delta_{fr} \end{bmatrix} F_{fr} + F_{rl} + F_{rr} + F_{\text{wind}} + G \tag{7}$$

where $\boldsymbol{T}$ is the coordinate transformation matrix, and F_{fl}, F_{fr}, F_{rl}, and F_{rr} denote the stresses from the four wheels, respectively. G is the gravity and F_{wind} is the resistance from air

$$F_{\text{wind}} = \begin{bmatrix} -f_{\text{wind}} & 0 & 0 \end{bmatrix}' \tag{8}$$

$$f_{\text{wind}} = C_w A v^2 g / 16 \tag{9}$$

where g is the acceleration of gravity, A is the front cross-sectional area of the vehicle, and v is the relative velocity of the vehicle with respect to the wind. C_w is a constant coefficient and set to be 0.35 in this paper.

Let $\boldsymbol{v}_l$ and $\boldsymbol{\psi}_{rb}$ denote the velocity in the local coordinates and the attitude angles in the road coordinates, respectively. The dynamics model of the vehicle can be described as follows:

$$\frac{d\boldsymbol{v}_l}{dt} = \boldsymbol{F}/m_c + \dot{\psi}_{rb} \times \boldsymbol{v}_l \tag{10}$$

where m_c is the total mass of the vehicle and F is the total force determined in (7).

C. MDP Modeling of the Overtaking Behavior

In a simple case, the state set of the MDP can be defined as $x = [l, v_a, v_f, v_1, d_1]$, where l denotes the current lane of the intelligent vehicle. If the intelligent vehicle is on the left lane, $l = 1$, and if the intelligent vehicle is fully on the right lane, $l = 2$. When the vehicle is changing from one lane to the other lane, the value of l is not changed until the lane changing process is completed. v_a and v_f denote the current velocity and the expected velocity of the intelligent vehicle during lane keeping, respectively. d_1 represents the relative distance between the intelligent vehicle and the nearest front vehicle, and v_1 stands for the velocity of the nearest front vehicle. As we have a general limitation of $d_{f\max} = d_1 = 0$, when there is no vehicle in the sensing range, we assume that $d_1 = d_{f\max}$ and $v_1 = v_a$. In the MDP model, we define two macro actions $\{a_1, a_2\}$ to accomplish an overtaking process, one action ($a_1 = 1$) is to keep the vehicle in or move to the right lane and the other action ($a_2 = 2$) is to keep the vehicle on or to move to the left lane.

We consider two kinds of reward models in this paper. The first is a combined model with one scalar reward function and the second is a model with multiple reward functions which stand for different objectives. The scalar reward function is defined as

$$r = \begin{cases} -300 & c = 1 \\ -150 & c = 0 \ \& \ z > Z \\ v_a - v_f - 0.1z - 0.2 & c = 0 \ \& \ z \le Z \ \& \ l = 2 \\ v_a - v_f - 0.1z & c = 0 \ \& \ z \le Z \ \& \ l = 1 \end{cases} \tag{11}$$

where z denotes the normalized value of acceleration, Z is a threshold for measuring the smoothness, and c is a flag which indicates a situation with a high possibility of collisions.

In the single reward function model, we consider safety, smoothness and speediness comprehensively. c is a flag for safety and the vehicle will get the minimum reward when $c = 1$. And when the safety objective is obtained, v_d will prevent the intelligent vehicle from changing the velocity abruptly. In addition, the third and the fourth conditions in (11) show that a higher velocity will get a higher reward as well.

Although the single reward model provides a solution to evaluate an overtaking process, it could not efficiently and flexibly realize the tradeoff among different objectives. Therefore, the MOMDP model has the following reward vector:

$$\vec{r} = \left\{ r^{(1)}, r^{(2)}, r^{(3)} \right\}$$
$$r^{(1)} = r^{\text{safe}} = \begin{cases} -1 & \text{if } c = 1 \\ 0 & \text{else} \end{cases}$$
$$r^{(2)} = r^{\text{speed}} = \begin{cases} v_a - v_f & v_a < v_f \ \& \ l = 1 \\ v_a - v_f - 0.2 & v_a < v_f \ \& \ l = 2 \\ 0 & v_a \ge v_f \ \& \ l = 1 \\ -0.2 & v_a \ge v_f \ \& \ l = 2 \end{cases}$$
$$r^{(3)} = r^{\text{smooth}} = \begin{cases} 0 \ , & z < Z \\ -z \ , & \text{else} \end{cases} \tag{12}$$

where r^{safe}, r^{speed}, and r^{smooth} represent evaluative feedbacks based on the objectives of safety, speediness, and smoothness, respectively. Since the three objectives can be optimized independently, we can adjust the priority of the three objectives to satisfy different driving preferences.

D. MORL Algorithm Based on API

Next, we will present the MO-API algorithm with data-driven feature learning for value function approximation. Suppose we have a total number of q objectives which correspond to a reward vector as $\vec{r} = \{r^1, r^2, \ldots, r^q\}$. As shown in Algorithm 1, the MO-API algorithm is designed to learn an approximation of the optimal action-value function $Q_i(x, a)$ for different objectives $r^i (i = 1, 2, \ldots, q)$. In MO-API, the action-value function $Q_i(x, a)$ $(i = 1, 2, \ldots, q)$ is approximated by a linear weighted combination of n basis functions

$$\hat{Q}_i(x, a) = \vec{\phi}^T(x, a)\vec{w}_i \tag{13}$$

where $\vec{w}_i = (w_1, w_2, \ldots, w_n)^T$ is the weight vector for objective i and $\vec{\phi}(x, a)$ is the feature vector, which is denoted by

$$\vec{\phi}(x, a) = (\phi_1(x, a), \phi_2(x, a), \ldots, \phi_n(x, a))^T. \tag{14}$$

In order to determine an appropriate set of feature vectors for value function approximation, feature learning approaches can be employed by making use of the samples. In MO-API, the first two steps implement the sampling and subsampling procedures, which obtain the sample set $D = \{(x_i, a_i, \vec{r}_i, x_i', a_i') \mid i = 1, 2, \ldots, m\}$ and a subset $D_s \subset D$, respectively. Based on the subset of samples, two strategies can be used for learn the features for value function approximation. One strategy is to use kernel-based feature learning for VFA and the other is manifold-based feature construction using graph Laplacian. For kernel-based feature learning, we adopt the sparse kernel learning method based on approximately linear dependence (ALD) analysis [31]. The ALD-based kernel sparsification approach implements an iterative process which constructs a kernel dictionary K_D incrementally. Suppose a subset of samples $D_s = \{x_i\}$ $(i = 1, 2, \ldots, n)$ are used to construct the kernel dictionary K_D. Initially, K_D only has one element $\{x_1\}$. The approximately linearly dependent condition of a new feature vector (x_t) is tested as follows [32]:

$$\delta_t = \min_c \left\| \sum_j c_j \phi(x_j) - \phi(x_t) \right\|^2 \le \mu \tag{15}$$

Algorithm 1 MO-API Algorithm

```
\\C: the number of clusters;
\\l: the number of basis functions.
1: Sampling:
   Using the vehicle model and the simulated environment to collect samples for learning. A set of samples $D = \{(s_i, a_i, \vec{r}_i, s_{i+1}\}$ is obtained by using an exploration policy.
2: Subsampling:
   Selecting a subset of samples $D_s$ using the $C$-means clustering method.
3: Based on the subset of samples $D_s$, perform feature learning for value function approximation.
4: For each objective $j$ ($j = 1, 2, \ldots, q$), learn a near-optimal policy using the collected sample set $\{(s_i, a_i, \vec{r}_i, s'_i, a'_i)\}$:
   Loop until a termination criterion is satisfied:
   (1) Policy evaluation:
       $w^{\pi_j} = \left(\Phi^T\left(\Phi - \gamma\Phi'\right)\right)^{-1}\Phi^T R^j$
   (2) Policy improvement:
       $\pi_j(s) = \arg\max_a \phi^T(s, a) w^{\pi_j}$
5: return a set of policies $\pi_j$ ($j = 1, 2, \ldots, q$) corresponding to the multiple objectives.
```

where $c = [c_j]$ and μ is a threshold parameter to determine the approximation accuracy and the sparsity level.

Another way to learn smooth features for value function approximation is to use the graph Laplacian approach, as studied in [32]. The graph Laplacian approach implements the feature learning process via the following two steps.

1) Constructing an undirected graph $G = (V, E, W)$ from the subset of samples D_s, where V is the vertex set consisting of all the elements in D_s, E is the edge set that connect each pair of vertex, and W is the weight matrix whose element w_{ij} are the weights of the edges.
2) Computing the graph Laplacian $L = D - W$ and calculating the m smoothest eigenvectors of L to form the feature matrix as $\vec{\phi}(x) = [\phi_1(x), \phi_2(x), \ldots, \phi_l(x)]^T$.

For a sample x that does not belong to the graph point set V, the feature vector can be computed by the Nystrõm extension method. For more details, please refer to [24]. To approximate the action-value functions of MDPs with continuous states and m discrete actions, the above basis functions can be repeated for each of the actions

$$\vec{\phi}(x, a) = \left[I(a, a_1)\vec{\phi}(x), I(a, a_2)\vec{\phi}(x), \ldots, I(a, a_n)\vec{\phi}(x)\right] \quad (16)$$

where $I(a, a_j)$ ($j = 1, 2, \ldots, m$) is an indicator function, i.e., if $a = a_j$, $I(a, a_j) = 1$, else $I(a, a_j) = 0$.

In MO-API, after learning the features for VFA, the following procedure is used to estimate the weight vector w_i ($j = 1, 2, \ldots, q$) of the optimal policy for each objective r_j ($j = 1, 2, \ldots, q$). Let

$$\Phi = \begin{pmatrix} \phi^T(x_1, a_1) \\ \phi^T(x_2, a_2) \\ \vdots \\ \phi^T(x_m, a_m) \end{pmatrix} \Phi' = \begin{pmatrix} \phi^T\left(x'_1, a'_1\right) \\ \phi^T\left(x'_2, a'_2\right) \\ \vdots \\ \phi^T\left(x'_m, a'_m\right) \end{pmatrix} R^i = \begin{pmatrix} r^i_1 \\ r^i_2 \\ \vdots \\ r^i_m \end{pmatrix} \quad (17)$$

where $D = \{(x_i, a_i, \vec{r}_i, x'_i, a'_i) \mid i = 1, 2, \ldots, m\}$ is a set of collected samples from an initial policy and $\vec{r}_i = \{r^1_i, r^2_i, \ldots, r^q_i\}$ is the reward vector defined for multiple objectives.

Then, the weight for approximating the action-value function and the corresponding policy based on the greedy strategy can be obtained as follows [22]:

$$\begin{cases} \vec{w}_i = \left(\Phi^T\left(\Phi - \gamma\Phi'\right)\right)^{-1}\Phi^T R^i \\ \pi_i(x) = \arg\max_a \phi^T(x, a)\vec{w}_i \end{cases} (i = 1, 2, \ldots, q). \quad (18)$$

Algorithm 2 DecisionFunction(x, p_j, π_j ($j = 1, 2, \ldots, q$)

```
Input: $x$—the current state; $p_j$—the weight for objective $j$; $\pi_j$—the policy learned for objective $j$. ($j = 1, 2, \ldots, q$)
1. For each objective $j$, compute the greedy near-optimal action:
   $a_j = \arg\max_a \vec{\phi}^T(x, a) w^{\pi_j}$ ($j = 1, 2, \ldots, q$)
   End for
2. Perform weighted voting to determine the final decision
   $a = u\left(\sum_{j=1}^{q} p_j a_j\right)$
   where $0 < p_j < 1$, $u(.)$ is a function that rounds the input to the nearest integers toward infinity.
Output: $a$
```

The output of Algorithm 1 includes multiple weight vectors which are used to approximate the value functions of different rewards from multiple objectives. Then, a multiobjective decision-making strategy will be used to choose the best actions by considering all the objectives comprehensively. Assume the m candidate actions are indexed by $1, 2, \ldots, m$. Algorithm 2 shows the decision function based on the learned multiple policies π_j ($j = 1, 2, \ldots, q$). The output a of Algorithm 2 is the index for the selected action.

The decision function described in Algorithm 2 uses the multiple policies learned by Algorithm 1 and a weighted voting strategy for multiobjective decision-making. By specifying different weights for each objective, we can realize different preferences for varying situations. For example, we can specify a larger weight for safety at first and determine whether to use a larger weight for speediness or smoothness. In the following simulation and experiments, two candidate weighting priorities were considered, which include the speed first priority (SPFP) which considered speediness first, and smoothness first priority (SMFP) which considered smoothness first.

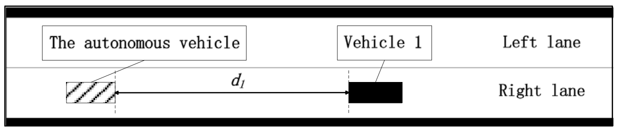


Fig. 6. Initial positions of the sampling process in the highway simulation environment.

IV. Simulation and Experimental Results

The performance of the proposed RL-based decision-making approach was evaluated both in simulation and experiments in this section. The samples were collected offline based on a highway simulation environment of the HQ-3 autonomous vehicle introduced above. Then, the RL-based decision module was trained using these samples and the learned policies were tested and compared comprehensively in the simulation environment. Furthermore, we also conducted real-time decision-making and overtaking control experiments on the autonomous HQ-3 vehicle to illustrate the effectiveness of the proposed method.

A. Simulation Environment

The simulations were carried out in a two-lane highway environment, as shown in Fig. 3. The width of each lane was set as $d = 5$ m, and the maximum range of the front sensors and the backward sensors were set as $d_{\text{front}} = 150$ m and $d_{\text{back}} = 100$ m, respectively. For RL methods, the discount factor γ is 0.95. The maximum number of iterations is set to be 1000 for all the algorithms.

In order to learn an appropriate overtaking strategy, the agent was first trained in a set of samples with 10 000 episodes. Let d_f denote the minimum distance of the front vehicles in the other lane and d_b denote the minimum distance of the backward vehicles. For a lane change behavior, a basic safe condition can be determined by comparing d_f, d_b with predefined safety thresholds D_1 and D_2. Then, in order to reduce the state and decision complexity, the basic safe condition for lane changing is determined by the following rule:

```
If $d_f > D_1$ and $d_b > D_2$
    Basic_Lane_Change_Condition = true
Else
    Basic_Lane_Change_Condition = False
End
```

Based on the above safety condition, in the sample-collection process, only the nearest front vehicle was considered, and the initial positions of both vehicles were located in the right lane, as shown in Fig. 6. Five measurement variables were chosen as the input of the RL-based decision module: $\{v_a, v_f, v_1, d_1, \text{action}\}$, where v_a is the current velocity of the intelligent vehicle, v_f is the expected velocity of the intelligent vehicle, v_1 is the current velocity of vehicle 1, d_1 is the distance between the intelligent vehicle and vehicle 1, and *action* is the current action taken by the intelligent vehicle. In the sampling process, one episode was defined as either the intelligent vehicle enters into a dangerous situation or a maximum sequence of 500 time steps has been simulated.

In this paper, we also designed an expert system with fixed rules to realize decision-making and prevent collisions in the traffic. The expert system considered the relative distances (d_1–d_4) and the velocities (v_1–v_4) of four nearest surrounding vehicles, as shown in Fig. 7. In addition, the normal velocity v_f during lane keeping was also considered in the expert system. In the following simulations, the expert system was not only used as the decision system for environmental vehicles but also used in the autonomous vehicle as a nonlearning method for performance comparison. The strategy definition of the expert system was recorded in Table I.

To test the performance in different kinds of traffic flow, we used a stochastic traffic model with a controllable traffic density. Here, "stochastic" means that the initial states of other vehicles in the traffic flow were generated randomly (with certain limitations). Fig. 8 showed a comparison of traffic models in different traffic densities. The density determined the current number of vehicles around the autonomous vehicle. The density of traffic flow was represented by μ, which determined the number of vehicles in the current 500-m range around the intelligent vehicle.

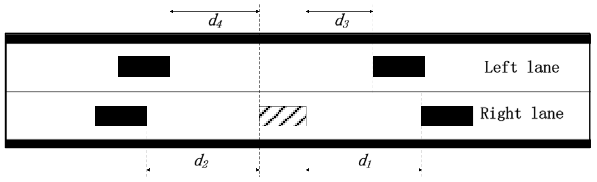


Fig. 7. Surrounding vehicles considered in the expert system.

TABLE I
Formulation of the Expert Decision Strategy

state	action	prerequisite
l=1 (right lane)	Move to the left lane	(d_1<50 | v_1-v_f<-3) & (d_3-d_1>10) & (v_3-v_1>1) & d_4>80
	Stay in the right lane	else
l=2 (left lane)	Stay in the left lane	((d_1>150 & v_1-v_f>1) | v_1>v_3) & d_2>60
	Move to the right lane	else

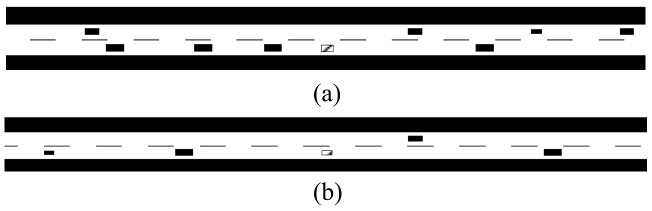


Fig. 8. Various traffic environments with different density values. "▨" represents the intelligent vehicle and "■" stands for other vehicles. (a) $\mu = 10$. (b) $\mu = 5$.

B. Comparison Between Learning-Based Decision-Making Methods and Traditional Rule-Based Methods

Learning-based decision-making methods can obtain optimized policies for different state conditions. However, it is difficult for traditional rule-based methods to be adaptive for dynamic situations. In this part, single-objective QL and a class of API algorithms called clustering-based RPI (C-RPI) were used to learn decision rules for the intelligent vehicle. The expert system with fixed rules defined in Table I was also evaluated. Simulations were conducted under a series of traffic density sequences ($\mu \in [5, 15]$).

In the simulation, one intelligent vehicle is simulated with different decision policies for performance testing. This intelligent vehicle is called the testing vehicle. Other intelligent vehicles are simulated with rule-based decision policies and these vehicles are called the environmental vehicles. There are several types of environmental vehicles which include cars, trucks, buses, etc. Different types of environmental vehicles have different expected velocities. In addition, for different traffic conditions, different numbers of environmental vehicles are generated.

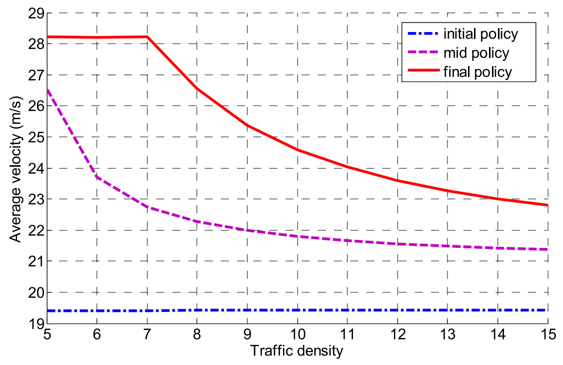
1) Performance of RL Methods After Different Iterations: In the following, the performance convergence of RL methods will be tested. Since C-RPI and API algorithms can be viewed as improved versions of batch mode QL with function approximation, we choose the tabular QL algorithm to learn driving policies and test the performance of different Q-tables after different number of learning iterations. Here we chose three iteration stages to generate the Q-tables respectively: the initial stage, the final stage and the "middle" stage. In the initial stage, the initial Q-table was randomly assigned with a range of [0, 100] and when QL was convergent, the final Q-table was obtained. The middle stage was defined as

$$e^{(k)} \leq \sqrt{e^{(0)}} \tag{19}$$

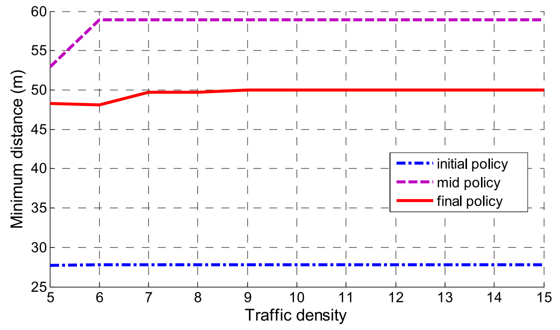
where $e^{(j)}$ ($j = 0, 1, \ldots, k$) denotes the summed temporal-difference error in iteration j.

The three Q-tables were tested in a traffic density sequence with a range of $5 \leq \mu \leq 15$. The comparison result was shown in Fig. 9. Fig. 9(a) showed that the overtaking policy based on the final Q-table achieved the highest average velocity while the policy from the initial Q-table obtained the lowest. The curve of the middle-stage Q-table dropped immediately with the increasing of traffic density, but the curve of the final Q-table approximately kept the same in a low traffic density ($5 \leq \mu \leq 7$), and the decline only appeared when μ exceeded 7. This suggested that the policy from the final Q-table was less affected by the change of traffic density. Fig. 9(b) showed the minimum distance between the intelligent vehicle and other vehicles in the driving process. Here the policy from the final Q-table achieved the relatively better result while the result from the initial Q-table is too close to the front vehicles to ensure the safety. Thus, as the learning process continued, the RL method can learn an optimized policy.

2) Comparison Between RL Methods and Expert System: In the following, we will make performance comparisons between two single-objective RL methods and the rule-based expert system. QL and C-RPI were tested as two RL methods which use tabular representations and value function approximation, respectively. First, we compared the reward curves of the three methods which were determined by the MDP model mentioned in Section III. In this simulation, the intelligent vehicle drove in a traffic flow with $\mu = 10$ for 300 steps. The reward



(a)



(b)

Fig. 9. Performance convergence of QL. Performance improvement of QL in terms of the (a) average velocity versus the traffic density and (b) minimum vehicle distance versus the traffic density.

for each step and the accumulated value were both recorded. In addition, we used the minimum relative distance, average velocity, and accumulation of velocity changes to describe the three objectives: 1) safety; 2) speediness; and 3) smoothness. The simulations were conducted in the traffic environment mentioned in Section IV-A with variable traffic densities. For each decision-making method, we repeated the tests for 50 times and the averaged values were used for comparison.

Fig. 10 shows the reward variations of the expert system, QL, and the C-RPI method. It is illustrated that C-RPI always obtain the highest reward value and the expert system remained the lowest. From Fig. 11(a), we can see that the intelligent vehicle using the policy learned by C-RPI has lower velocities than the policy obtained by QL, while higher than the policy obtained by the expert system. The accumulation of velocity changes showed an opposite situation. In Fig. 11(c), we found C-RPI obtained more appropriate vehicle distances (around 50 m) than the other two methods in traffic densities ranging from 5 to 11. In traffic conditions with higher traffic densities, both C-RPI and expert rules prefer larger vehicle distances than the policy obtained by QL.

Although the expert system method obtained the minimum accumulation of velocity changes in most situations, it realized the least number of overtaking actions, which is not suitable for the speediness objective.

3) Performance Testing in Multilane Highway Simulation Environment: In order to test the performance of the lane changing policy learned by the proposed RL method, a multilane highway simulation environment was also designed. The three-lane highway environment model is shown in Fig. 12, which includes a left lane, a middle lane, and a right lane.

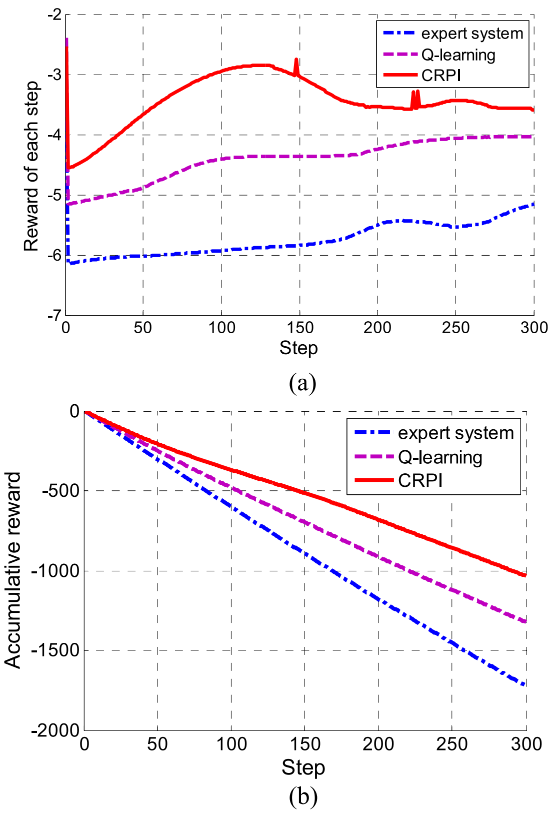


Fig. 10. Reward comparison between RL methods and the expert system. Performance comparisons of different methods in terms of the (a) reward at each step and (b) accumulative reward.

The initial position of the autonomous vehicle was set to the right lane. The environmental vehicles were generated with different types such as cars, trucks and buses randomly. The initial states of these vehicles (including relative distances and velocities) were generated randomly within certain limits. In addition, the expected velocities of these vehicles were set based on the vehicle's type and were changed according to the traffic conditions. The vehicle dynamics model and control characteristics shown in Section III-B were also considered in the simulation process.

The MDP state s of the decision-making module includes the current lane l and the longitudinal speed v_0 of the testing vehicle, the velocities and distances of other environmental vehicles. It can be formally defined as

$$s = [l, v_0, v_1, d_1, v_2, d_2, v_3, d_3, v_4, d_4, v_5, d_5, v_6, d_6]. \quad (20)$$

The action a of the MDP determines the target lane of the testing intelligent vehicle. Since the highway environment has three lanes, the action includes three elements which determine the right lane, the middle lane and the left lane as the target lane of the testing vehicle, respectively. The action set of the MDP is formally defined as follows:

$$A = [1, 2, 3]. \quad (21)$$

The reward function incorporates the requirements of safety, the expected speed and traffic rules. In addition, the speediness of the testing vehicle is also considered. Define c as a flag which indicates a situation with a high possibility of collisions. Then the reward function is defined as follows (d_s is the

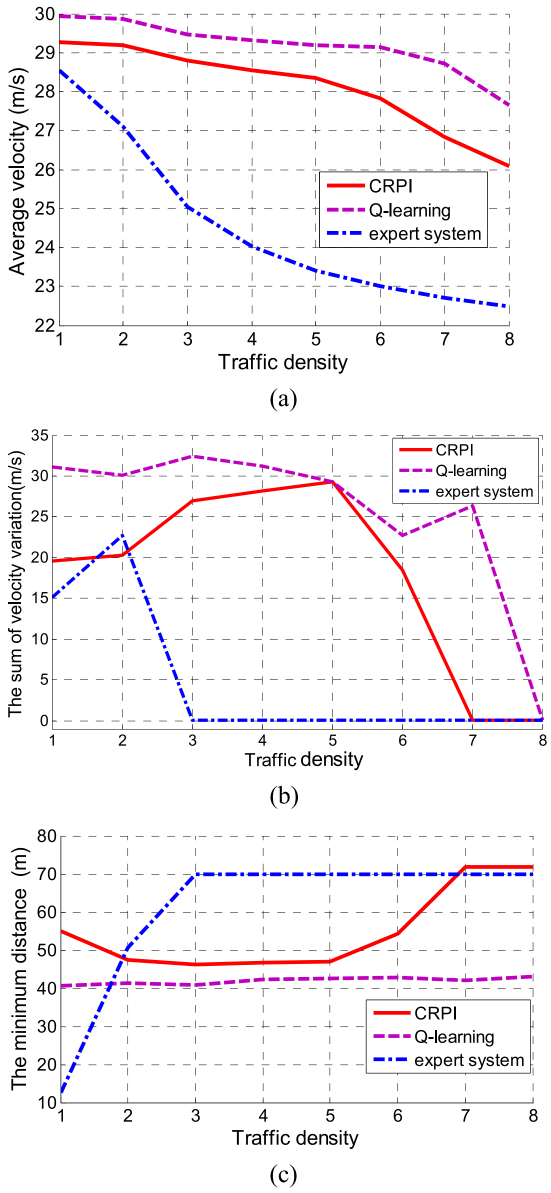


Fig. 11. Performance comparisons among different methods. (a) Average velocity. (b) Variations of velocity changes. (c) Minimum distance.

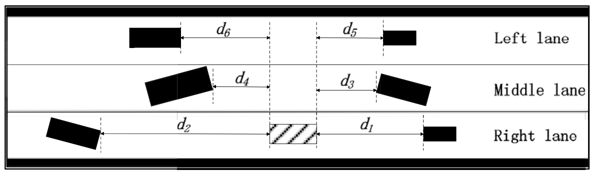


Fig. 12. Schematic of the three-lane highway modeling.

safe distance):

$$r = \begin{cases} -1000 & c = 1 \\ -10 & |l - l'| > 1 \\ -10 & c = 0, l = 1, v_0 < 60 \text{ km/h} \\ -5 & c = 0, l = 2, v_0 < 90 \text{ km/h} \\ -5 & c = 0, l = 3, v_0 < 110 \text{ km/h} \\ -5 & c = 0, d_1 = d_S, l \neq l' \\ -|v_0 - v_{\text{free}}| & c = 0. \end{cases} \quad (22)$$

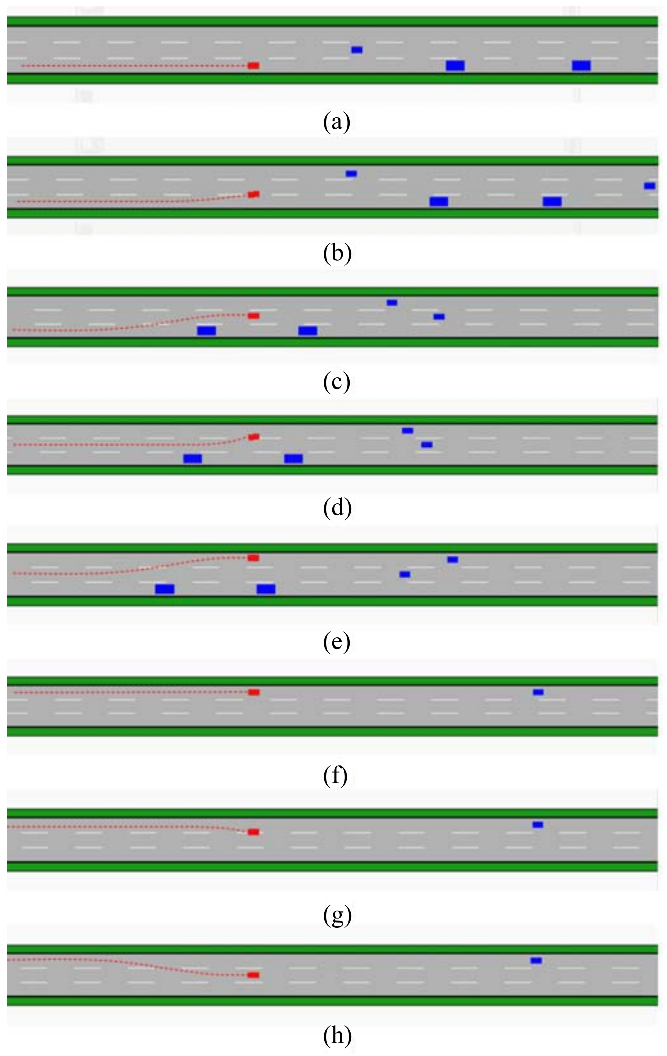


Fig. 13. Driving decision test in a three-lane traffic environment (red for the autonomous vehicle and blue for environmental vehicles). The subplots (a) to (h) show several successive lane-changing and overtaking behaviors of the intelligent vehicle in a multi-lane highway environment.

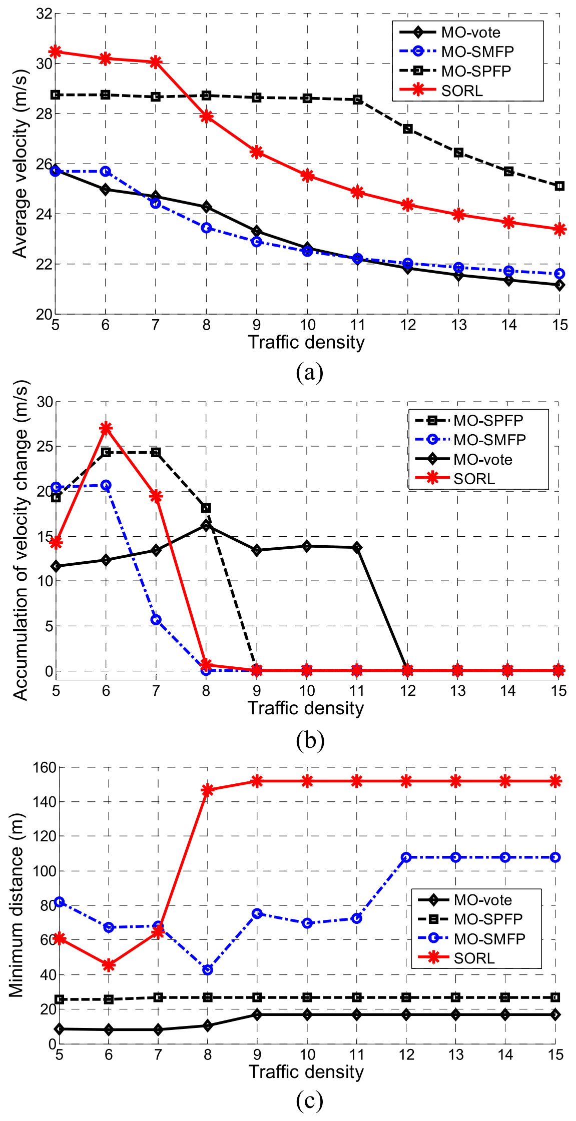


Fig. 14. Performance statistics of MORL and SORL. (a) Average velocities. (b) Variations of velocity changes. (c) Minimum distances.

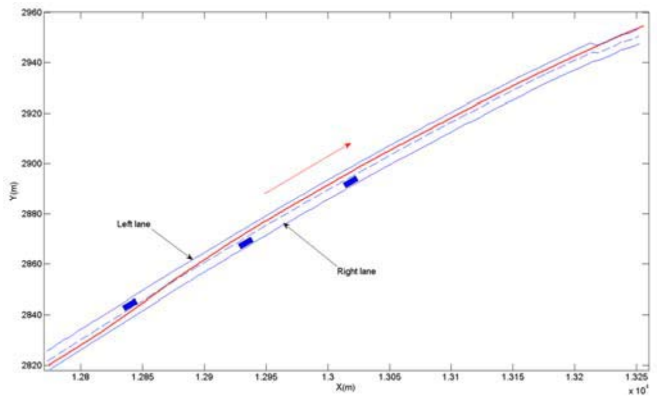


Fig. 15. Trajectory of a successful overtaking behavior (red line: the vehicle trajectory and blue line: the detected lane boundaries.

Before the learning process, 12 000 samples were collected by using a random action policy. For the API algorithm, we used the kernel-based feature learning method to approximate the value function. In particular, the multikernel feature learning approach introduced in [43] was employed in the MO-API algorithm. Three Gaussian kernels with different width parameters were combined to approximate the value functions. The maximum iteration number was set as 20, and the termination condition for iteration err or is 10^{-5}. After a series of iterations, an optimized policy can be obtained for performance testing.

Fig. 13 shows the decision outputs of the testing vehicle in the simulation environment. Fig. 13(a)–(c) shows that the testing vehicle overtook the front vehicle by changing from the right lane to the middle lane. Fig. 13(d)–(f) shows that the testing vehicle overtakes other vehicles in the middle lane. Fig. 13(g) and Fig. 13(h) illustrate that the testing vehicle changes back to the middle lane. When the testing vehicle satisfies the condition for changing lanes and the velocity is lower than the expected velocity, it will change its lane and accelerate to the expected velocity. Otherwise, the intelligent vehicle will follow the current lane. In addition, the intelligent vehicle will change to the middle lane or the right lane when surrounding vehicles do not affect driving in order to facilitate other vehicles overtaking.

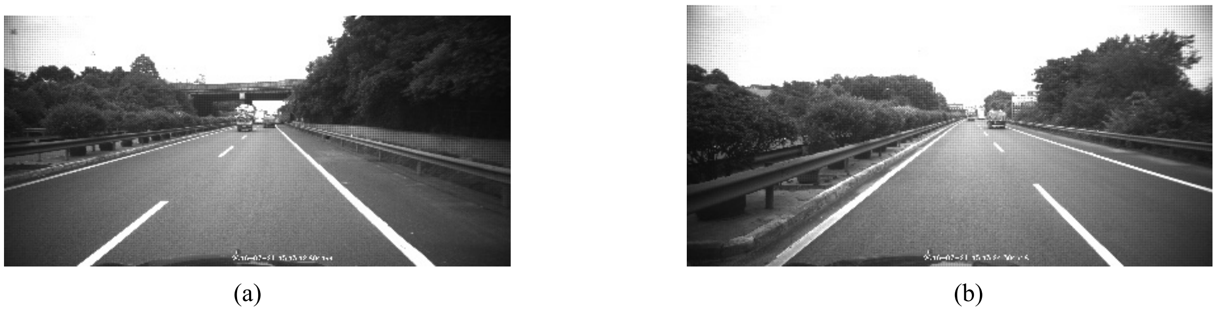


Fig. 16. Front views from the autonomous vehicle during the left lane-change behavior. (a) Front view at the starting point for lane change. (b) Front view at the final stage for left lane-change.

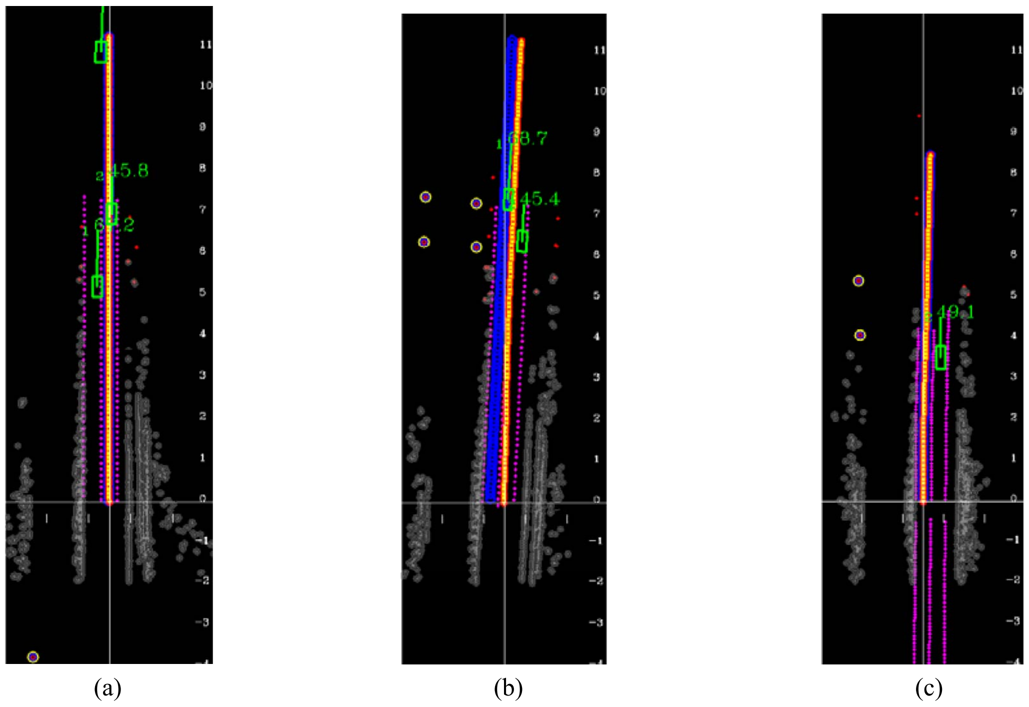


Fig. 17. Decision and planning outputs during the left lane-change behavior. (a) Starting left lane-change. (b) Middle stage of left lane change. (c) Final point of left lane-change.

C. Comparison Between SORL Method and MORL Method

In autonomous driving, several related objectives should be considered, including the safety in heavy traffic and speediness in light traffic, and smoothness was also important for human passengers. In this section, the MO-API method was compared with SORL methods, and the performance of MO-API with different priorities of objectives was also studied.

There were several strategies for solving MORL problems. For MORL, the voting strategy is a basic approach to realize a compromise among the three objectives, which was also named the equal priority (EP) voting. In order to express flexible priorities, the weighted voting method was used in this paper. We defined two objective-priorities instead of the EP: 1) the SPFP which considered speediness first and 2) SMFP which considered smoothness first.

In both SPFP and SMFP, safety should always have a higher priority. Therefore we defined the weight of safety as 0.4 which was equal to the weight of the objective considered first, and the weight of the rest objective was 0.2. In addition, optimized policies for the single-objective MDP model and MO-MDP model were both learned by the MO-API approach with C-RPI. We repeated the simulation tests for 50 times. The statistical results were shown in Fig. 14.

D. Real-Time Experiments on the HQ-3 Autonomous Vehicle

In order to test the effectiveness of the MO-API approach for autonomous decision making, we also performed real-time decision making experiments on an autonomous vehicle in highway environments with normal traffic. The HQ-3 autonomous vehicle is equipped with multiple sensors including cameras, laser radars, microwave radars, etc. In our experiments, the decision making system receives the local map generated by the sensing and mapping system and outputs the decision for lane changing or lane keeping. Training data were collected by making use of the simulation environment described in Section III.

As shown in Fig. 7, the state vector at each time step is defined as $s = [l, v_0, v_1, d_1, v_2, d_2, v_3, d_3, v_4, d_4]$, where l

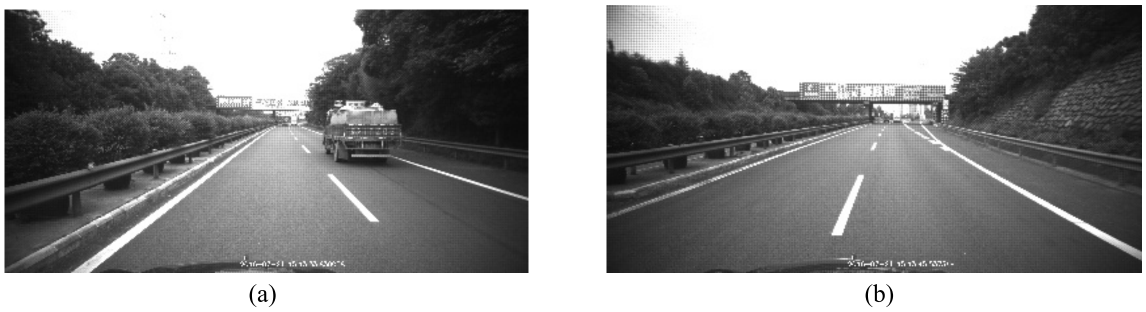


Fig. 18. Front views from the autonomous vehicle during the left lane-change behavior. Front view at the (a) starting point for right lane change and (b) final stage for right lane-change.

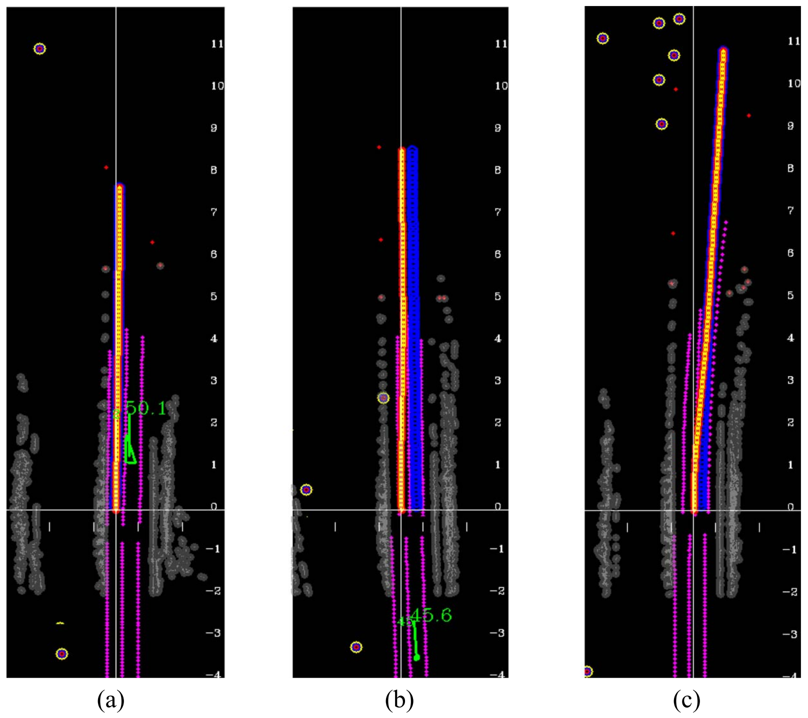


Fig. 19. Decision and planning outputs during the right lane-change behavior. (a) Starting point. (b) Middle stage. (c) Final point.

is the current lane number of the autonomous vehicle (for left lane, $l = 1$ and for right lane $l = 2$), v_0 is the longitudinal velocity of the autonomous vehicle, v_1, d_1 are the velocity and distance of the nearest front vehicle in the right lane, respectively, v_2, d_2 are the velocity and distance of the nearest rear vehicle in the right lane, respectively, v_3, d_3 are the velocity and distance of the nearest front vehicle in the left lane, respectively, and v_4, d_4 denote the velocity and distance of the nearest rear vehicle in the left lane, respectively. The MO-API algorithm uses the simulated sample data $D = \{(x_i, a_i, \vec{r}_i, x_i', a_i') \mid i = 1, 2, \ldots, m\}$ to learn a set of optimized decision policies which correspond to different objectives. The aim of the experimental validation is focused on the real-time performance testing of the learned policies using the RL approach based on API. Therefore, in our implementation, a combined scalar reward function is defined as follows: if the collision probability is higher than a threshold, $r = -1000$, otherwise if the vehicle is running on the left lane, $r = -0.5$, otherwise $r = -\|v_0 - v_{\exp}\|$, where $v_{\exp}$ is an expected velocity.

The sample collection process uses a random decision policy and uniform distributions to generate samples. In each episode, the velocities of different vehicles were initialized within the interval [40, 100 km/h], the distances between the autonomous vehicle and other front vehicles were initialized within the interval [10, 150 m], and the distances between the autonomous vehicle and other rear vehicles were initialized within the interval [10, 100 m]. The samples were collected by simulating the autonomous vehicle either on the left lane or on the right lane. For each case, 45 000 state transition samples were observed. So, the total number of samples is 90 000. The feature vector for value function approximation in API is generated from the ALD-based kernel sparsification process, which is described in (14). Gaussian kernels are used for kernel-based feature representation. The width for Gaussian kernels was selected as $\sigma = 100$. The parameters used for kernel sparsification is $\mu = 0.4$.

After the convergence of the API algorithm, an optimized decision policy was obtained and the policy was implemented in the real-time decision system of the HQ-3 autonomous vehicle. The autonomous driving system of HQ-3 was tested on a two-lane highway environment with normal traffic. Fig. 15 shows the trajectory of a complete lane-change and overtaking decision and control process, where several other vehicles running on different lanes were detected. It is shown that the autonomous vehicle can perform lane-change and overtaking behaviors successfully and smoothly. Fig. 16(a) and (b) shows the front views from the autonomous vehicles at the beginning stage and final stage of a left lane-change behavior, respectively.

Fig. 17 shows the decision and planning outputs of the decision system, where the local sensing map was also illustrated. The detected lanes are shown in dotted lines. The rectangles with arrows illustrate the detected vehicles on the lanes, where the arrows indicate the estimated moving directions of the detected vehicles. Fig. 18(a) and (b) shows the front views from the autonomous vehicles at the beginning stage and final stage of a right lane-change behavior, respectively. The decision and planning outputs of the decision system were also illustrated in Fig. 19. As shown in Fig. 19, based on the decision outputs of the learned policy, the planning module can successfully generate safe, and smooth paths for the intelligent vehicle.

V. Conclusion

In this paper, an RL approach with value function approximation and feature learning was proposed for driving decision-making of intelligent vehicles. The driving decision making problem was modeled as an MDP and an MO-API algorithm was presented. Compared with previous MORL approaches, the proposed approach used API and value function approximation with feature learning to achieve better learning efficiency. Two feature learning strategies have been introduced for MO-API. In simulations, a 14-DOF vehicle model and stochastic traffic flow models were used to generate training samples and testing the performance of different decision-making methods. The results have demonstrated the effectiveness of the proposed approach. Real-time decision-making experiments were also performed for testing the effectiveness of the RL-based approach. Although the simulation and experimental results illustrate that the proposed RL approach can obtain appropriate decision making policies in different traffic conditions, there are still more works that needs further investigations in the future. One interesting problem is to combine other adaptive dynamic programming and RL methods [36]–[41] to realize optimized trajectory generation and tracking control. The other one is to extend the RL-based decision making approach to more complex traffic conditions.

Acknowledgment

The authors would like to thank the Associate Editor and the anonymous reviewers for their valuable comments and suggestions, which greatly improved the quality of this paper.

References

[1] A. Niehaus and R.F. Stengel, "Probability-based decision making for automated highway driving," *IEEE Trans. Veh. Technol.*, vol. 43, no. 3, pp. 626–634, Aug. 1994.

[2] D. Bevly *et al.*, "Lane change and merge maneuvers for connected and automated vehicles: A survey," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 105–120, Mar. 2016.

[3] D. Zhao *et al.*, "Accelerated evaluation of automated vehicles safety in lane-change scenarios based on importance sampling techniques," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 3, pp. 595–607, Mar. 2017.

[4] R. Schubert, K. Schulze, and G. Wanielik, "Situation assessment for automatic lane-change maneuvers," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 3, pp. 607–616, Sep. 2010.

[5] M. Ardelt, C. Coester, and N. Kaempchen, "Highly automated driving on freeways in real traffic using a probabilistic framework," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 4, pp. 1576–1585, Dec. 2012.

[6] A. Eskandarian, Ed., *Handbook of Intelligent Vehicles*. London, U.K.: Springer-Verlag, 2012.

[7] R. Schubert, "Evaluating the utility of driving: Toward automated decision making under uncertainty," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 1, pp. 354–364, Mar. 2012.

[8] C. Urmson *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *J. Field Robot.*, vol. 25, no. 8, pp. 425–466, 2008.

[9] S. Ulbrich and M. Maurer, "Probabilistic online POMDP decision making for lane changes in fully automated driving," in *Proc. 16th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, 2013, pp. 2063–2067.

[10] C. Kim and R. Langari, "Adaptive analytic hierarchy process-based decision making to enhance vehicle autonomy," *IEEE Trans. Veh. Technol.*, vol. 61, no. 7, pp. 3321–3332, Sep. 2012.

[11] C. R. Baker and J. M. Dolan, "Traffic interaction in the urban challenge: Putting boss on its best behavior," in *Proc. Int. Conf. Intell. Robots Syst.*, 2008, pp. 1752–1758.

[12] J. Wei and J. M. Dolan, "A robust autonomous freeway driving algorithm," in *Proc. IEEE Intell. Veh. Symp.*, 2009, pp. 1015–1020.

[13] J. Wei, J. M. Dolan, and B. Litkouhi, "A prediction- and cost function-based algorithm for robust autonomous freeway driving," in *Proc. IEEE Intell. Veh. Symp.*, San Diego, CA, USA, 2010, pp. 512–517.

[14] J. Pérez, V. Milanés, E. Onieva, J. Godoy, and J. Alonso, "Longitudinal fuzzy control for autonomous overtaking," in *Proc. IEEE Int. Conf. Mechatronics*, Istanbul, Turkey, 2011, pp. 188–193.

[15] R. Schubert and G. Wanielik, "A unified Bayesian approach for object and situation assessment," *IEEE Intell. Transp. Syst. Mag.*, vol. 3, no. 2, pp. 6–19, Jun. 2011.

[16] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[17] L. Busoniu, R. Babuska, B. D. Schutter, and D. Ernst, *Reinforcement Learning and Dynamic Programming Using Function Approximators*. Boca Raton, FL, USA: CRC Press, 2010.

[18] D. C. K. Ngai and N. H. C. Yung, "Automated vehicle overtaking based on a multiple-goal reinforcement learning framework," in *Proc. IEEE Intell. Transp. Syst. Conf.*, Seattle, WA, USA, 2007, pp. 818–823.

[19] 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.

[20] 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. (ICMLC)*, 2013, pp. 362–369.

[21] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *J. Mach. Learn. Res.*, vol. 4, pp. 1107–1149, Jan. 2003.

[22] X. Xu, Z. Huang, D. Graves, and W. Pedrycz, "A clustering-based graph Laplacian framework for value function approximation in reinforcement learning," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2613–2625, Dec. 2014.

[23] S. Doltsinis, P. Ferreira, and N. Lohse, "An MDP model-based reinforcement learning approach for production station ramp-up optimization: Q-learning analysis," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 44, no. 9, pp. 1125–1138, Sep. 2014.

[24] N. Wesner, "Multi-objective optimization via visualization," *Econ. Bull.*, vol. 37, no. 2, pp. 1226–1233, 2017.

[25] S. Ruzika and M. M. Wiecek, "Approximation methods in multiobjective programming," *J. Optim. Theory Appl.*, vol. 126, no. 3, pp. 473–501, 2005.

[26] T. Erfani and S. V. Utyuzhnikov, "Directed search domain: A method for even generation of the Pareto frontier in multiobjective optimization," *J. Eng. Optim.*, vol. 43, no. 5, pp. 467–484, 2011.

[27] I. Giagkiozis and P. J. Fleming, "Methods for multi-objective optimization: An analysis," *Inf. Sci.*, vol. 293, pp. 338–350, Feb. 2015.

[28] P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, and E. Dekker, "Empirical evaluation methods for multiobjective reinforcement learning algorithms," *Mach. Learn.*, vol. 84, nos. 1–2, pp. 51–80, 2011.

[29] C. Liu, X. Xu, and D. Hu, “Multiobjective reinforcement learning: A comprehensive overview,” *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 3, pp. 385–398, Mar. 2015.
[30] R. A. Howard, *Dynamic Programming and Markov Processes*. Cambridge, MA, USA: MIT Press, 1960.
[31] 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.
[32] S. Mahadevan and M. Maggioni, “Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes,” *J. Mach. Learn. Res.*, vol. 8, pp. 2169–2231, Jan. 2007.
[33] S. J. Bradtke and A. G. Barto, “Linear least-squares algorithms for temporal difference learning,” *Mach. Learn.*, vol. 22, nos. 1–3, pp. 33–57, 1996.
[34] J. Johns, M. Petrik, and S. Mahadevan, “Hybrid least-squares algorithms for approximate policy evaluation,” *Mach. Learn.*, vol. 76, nos. 2–3, pp. 243–256, 2009.
[35] Z. Huang, X. Xu, and L. Zuo, “Reinforcement learning with automatic basis construction based on isometric feature mapping,” *Inf. Sci.*, vol. 286, pp. 209–227, Dec. 2014.
[36] D. Liu, Q. Wei, and P. Yan, “Generalized policy iteration adaptive dynamic programming for discrete-time nonlinear systems,” *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 12, pp. 1577–1591, Dec. 2015.
[37] H. G. Zhang, D. R. Liu, Y. H. Luo, and D. Wang, *Adaptive Dynamic Programming for Control: Algorithms and Stability*. London, U.K.: Springer, 2013.
[38] F. L. Lewis and D. Vrabie, “Reinforcement learning and adaptive dynamic programming for feedback control,” *IEEE Circuits Syst. Mag.*, vol. 9, no. 3, pp. 32–50, 3rd Quart., 2009.
[39] T. Dierks and S. Jagannathan, “Online optimal control of affine nonlinear discrete-time systems with unknown internal dynamics by using time-based policy update,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 7, pp. 1118–1129, Jul. 2012.
[40] H. Xu, Q. Zhao, and S. Jagannathan, “Finite-horizon near-optimal output feedback neural network control of quantized nonlinear discrete-time systems with input constraint,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 8, pp. 1776–1788, Aug. 2015.
[41] H. Kebriaei, A. Rahimi-Kian, and M. N. Ahmadabadi, “Model-based and learning-based decision making in incomplete information cournot games: A state estimation approach,” *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 4, pp. 713–718, Apr. 2015.
[42] X. Li, X. Xu, and L. Zuo, “Reinforcement learning based overtaking decision-making for highway autonomous driving,” in *Proc. 6th Int. Conf. Intell. Control Inf. Process. (ICICIP)*, 2015, pp. 336–342.
[43] J. Liu, X. Xu, Z. Huang, and C. Lian, “Model-free multi-kernel learning control for nonlinear discrete-time systems,” *Int. J. Robot. Autom.*, vol. 32, no. 5, pp. 538–550, 2017.

Xin Xu (M’07–SM’12) received the B.S. degree in electrical engineering from the Department of Automatic Control, National University of Defense Technology (NUDT), Changsha, China, in 1996, where he received 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 in Hong Kong Polytechnic University, University of Alberta, University of Guelph, and the University of Strathclyde, U.K., respectively. He is currently a Professor with the College of Mechatronics and Automation, NUDT, China. He has co-authored more than 160 papers in international journals and conferences, and co-authored four books. His research interests include intelligent control, reinforcement learning, approximate dynamic programming, machine learning, robotics, and autonomous vehicles.

Dr. Xu received the Fork Ying Tong Youth Teacher Fund of China in 2008 and the 2nd class National Natural Science Award of China in 2012. He serves as the co-Editor-in-Chief of *Journal of Intelligent Learning Systems and Applications*. 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 International Journal of Social Robotics*. He 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.

Lei Zuo received the Bachelor’s degree in electrical engineering and the Ph.D. degree in automation both from the College of Mechatronics and Automation, National University of Defense Technology, China, in 2009 and 2016, respectively.

He is currently a Lecturer with the College of Electronics Technology, National University of Defense Technology, China. His research interests include reinforcement learning, approximate dynamic programming (ADP), and autonomous vehicles. He has coauthored more than 10 papers in international journals and conferences.

Xin Li received the Master’s degree in automation from the College of Mechatronics and Automation (CMA), NUDT, China, in 2015. He is currently working toward the Ph.D. degree at the Institute of Navigation Technology, CMA, NUDT. His research interests focus on computer vision and unmanned aerial vehicles.

Lilin Qian received both the Bachelor’s degree and Master’s degree in electrical engineering and automation from Aviation University Air Force, China, in 2012 and 2015, respectively. He is currently working toward the Ph.D. degree at the Institute of Unmanned Systems, College of Artificial Intelligence, NUDT, China.

His research interests include reinforcement learning, trajectory planning, behavior decision, and autonomous vehicles.

Junkai Ren received the Master’s degree in control science and engineering from the College of Mechatronics and Automation, National University of Defense Technology (NUDT), China, in 2017. He is currently working toward the Ph.D. degree from the Institute of Unmanned Systems, NUDT. His research interests include machine learning, robot control, and autonomous vehicles.

Zhenping Sun received the Ph.D. degree in pattern recognition and intelligent system from the College of Mechatronic Engineering and Automation, National University of Defense Technology, Changsha, China, in 2004.

He is currently an Associate Professor with the National University of Defense Technology. His research activities include the hardware and software architecture design for autonomous ground vehicles. His research interests include robotic motion planning and intelligent control of autonomous vehicles.