An Approach to Tune Fuzzy Controllers Based on Reinforcement Learning for Autonomous Vehicle Control

Xiaohui Dai, Chi-Kwong Li, *Senior Member, IEEE*, and A. B. Rad, *Senior Member, IEEE*

***Abstract*—In this paper, we suggest a new approach for tuning parameters of fuzzy controllers based on reinforcement learning. The architecture of the proposed approach is comprised of a Q estimator network (QEN) and a Takagi–Sugeno-type fuzzy inference system (TSK-FIS). Unlike other fuzzy Q-learning approaches that select an optimal action based on finite discrete actions, the proposed controller obtains the control output directly from TSK-FIS. With the proposed architecture, the learning algorithms for all the parameters of the QEN and the FIS are developed based on the temporal-difference (TD) methods as well as the gradient-descent algorithm. The performance of the proposed design technique is illustrated by simulation studies of a vehicle longitudinal-control system.**

***Index Terms*—Autonomous vehicles, fuzzy controllers, longitudinal control, reinforcement learning.**

I. Introduction

FUZZY logic control as a paradigm to deal with unknown system dynamics and uncertainty of environment has achieved success in many practical systems such as subway systems, nuclear reactor control, and automobile transmission control, etc. [1]. However, the performance of a fuzzy controller depends on its rule base, which may be difficult to compile. Several approaches have been presented to learn and tune the fuzzy rules to achieve the desired performance. Some popular approaches design the fuzzy systems from input–output data based on the following methods: 1) a table look-up scheme; 2) gradient-descent training; 3) recursive least square; and 4) clustering, etc. [2]. Unfortunately, the input–output data may not be readily available, and the above methods may belong to supervised learning methods. Another widely used method is to implement adaptive laws based on the Lyapunov synthesis approach [3]. However, a disadvantage of adaptive fuzzy control is that it requires some model information of the system and it may only aim at a specific class of systems.

Manuscript received January 26, 2004; revised December 8, 2004, January 11, 2005, and January 20, 2005. This work was supported by the Hong Kong Polytechnic University under Research Grant G-W085. The Associate Editor for this paper was R. Goudy.

X. Dai is with the Rockwell Automation Research Center, Shanghai 2002333, China (e-mail: dxdai@ra.rockwell.com).

C.-K. Li is with Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong (e-mail: enckli@polyu.edu.hk).

A. B. Rad is with Department of Electrical Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong (e-mail: eeabrad@polyu.edu.hk).

Digital Object Identifier 10.1109/TITS.2005.853698

Reinforcement learning has attracted considerable attention in the past because it provides an effective approach to control and decision problems for which optimal solutions are analytically unavailable or difficult to obtain. Reinforcement learning is based on the common-sense idea that if an action is followed by a satisfactory state, or by an improvement, then the tendency to produce that action is strengthened, i.e., reinforced. In essence, reinforcement learning is a direct adaptive optimal control [4]. Unlike the supervised learning, the correct actions or trajectories are not required for reinforcement learning.

For the above reasons, reinforcement learning has been applied to tune the fuzzy rules of fuzzy systems. It is shown that supervised learning is more efficient than reinforcement learning [14]. However, reinforcement learning only needs the critic information, i.e., rewards and punishments (evaluative signal) with respect to the different states of the controlled system [5]. The actual correct actions or their trajectories are not needed. Since the evaluative signal contains much less information than the reference signal, the reinforcement learning is appropriate for systems operating in a knowledge-poor environment [6].

We employ a Takagi–Sugeno-type fuzzy system as the controller of the system. The parameters of this fuzzy controller are tuned based on reinforcement learning with the "evaluative signal" instead of the "teacher" signal. From the point of view of reinforcement learning, a fuzzy inference system (FIS) is a means to introduce generalization in the state space and generate continuous actions in the reinforcement-learning problem whereas from the point of view of FISs, reinforcement learning is a learning method used to tune a fuzzy controller in a flexible way.

However, unlike many fuzzy Q-learning approaches that select the optimal action based on finite discrete actions [21], [34], [35], we obtain a continuous control output from Takagi–Sugeno-type (TSK)-FIS.

This paper is organized as follows: Section II briefly introduces reinforcement learning. The architecture of the controller is described in Section III. The learning algorithms and parameter update laws are presented in Section IV. Section V illustrates the performance of our proposed method through vehicle navigation. Finally, conclusions are drawn in Section VI.

II. Reinforcement Learning

In reinforcement learning, the system is instructed indirectly about the performance of the current control action through an evaluation signal. The study of reinforcement learning relates to the credit assignment where, given the performance of a process, one has to assign the reward or punishment attributed to individual elements that contribute to that performance. Temporal-difference (TD) methods can be used to solve the temporal credit-assignment problem [5].

The application of reinforcement learning in control problems focuses on two main types of algorithms: actor–critic learning and Q-learning. The actor–critic learning system contains two parts: one to estimate the state value function $V(x)$, and the other to choose the optimal action for each state. For Q-learning, the system estimates an action value function $Q(x,a)$ for all state-action pairs, and selects the optimal-control algorithm based on $Q(x,a)$.

The state value function $V(x)$ is the expected discounted sum of rewards with the initial state x, and can be written as

$$V(x) = E\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | x_t = x\right\} \tag{1}$$

where x is the state of the system, r_{t+k+1} is the instant reward at time $t+k+1$, $\gamma \in [0,1]$ is a discount factor, and $E(\cdot)$ is the expected value function.

The action value function $Q(x,a)$ is the expected discounted sum of rewards with the initial state x and initial action a, and can be written as

$$Q(x,a) = E\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | x_t = x, a_t = a\right\} \tag{2}$$

where a is the action that acts on the system. The optimal-state value function $V^*(x)$ and the optimal-action value function $Q^*(x,a)$ are represented as

$$V^*(x) = \max_a \{r(x_{t+1}) + \gamma V^*(x_{t+1})\}$$
$$Q^*(x,a) = E\left\{r(x_{t+1}) + \gamma \max_{a'} Q^*(x_{t+1}, a') | x_t = x, a_t = a\right\}.$$

The Q-learning estimates the $Q^*(x,a)$ based on TD error δ_t

$$\delta_t = r_{t+1} + \gamma \max_{a'} Q(x_{t+1}, a') - Q(x_t, a_t) \tag{3}$$

and $Q(x,a)$ is updated according to the above TD error as

$$Q_{t+1}(x_t, a_t) = Q_t(x_t, a_t) + \alpha\delta_t \tag{4}$$

where α is the learning rate. Watkins [7] has shown that $Q_t(x,a)$ converges to $Q^*(x,a)$ with a probability 1 if all actions continue to be tried from all states, and the following conditions for α are satisfied:

$$0 < \alpha_k < 1, \ \sum_{k=1}^{\infty} \alpha_k = \infty \text{ and } \sum_{k=1}^{\infty} \alpha_k^2 < \infty.$$

In fact, the definition of the above TD error is TD(0), and can be extended to $\mathrm{TD}(\lambda)$, which considers the past and the current estimated error using eligibility traces e_t:

$$e_t = \sum_{k=0}^{t} (\lambda\gamma)^{t-k} \chi_x(k) \tag{5}$$

where λ is the recency factor (also known as eligibility rate) and $0 \le \lambda \le 1$, $\chi_x(t) = \begin{cases} 1, & x_t = x \\ 0, & \text{otherwise.} \end{cases}$

Then, $Q(x,a)$ is updated according to following rule:

$$Q_{t+1}(x_t, a_t) = Q_t(x_t, a_t) + \alpha\delta_t e_t. \tag{6}$$

After obtaining the estimate of the optimal-action value function $Q^*(x,a)$, we can obtain the optimal action a^*, which maximizes $Q^*(x,a)$ for the current state x, i.e., $a^* = \arg_a \max Q^*(x,a)$. However, during the process of estimating $Q^*(x,a)$, we should use a stochastic control rule that will take all possible actions to satisfy the requirements of Watkins' theorem. The Boltzmann exploration and ε-greedy policy are such stochastic control rules, which are often adopted [5], [10].

The reinforcement learning was proposed to deal with discrete states and discrete actions originally based on Markovian decision problems (MDP) [5]. In the case of a large continuous state space, the discrete representation of reinforcement learning is intractable. This problem is known as the curse of dimensionality. To solve this problem and generalize from previously experienced states to ones that have never been seen, we should combine the reinforcement learning with existing generalization methods, such as a low-order polynomial, neural network, or fuzzy system instead of a look-up table. In others words, we can estimate the optimal-state value function $V^*(x)$ or the optimal-action value function $Q^*(x,a)$ with approximators.

A number of successful applications of reinforcement learning have been reported, such as board games [13], mountain car [22], inverted pendulum [20], and robot navigation [25], etc.

We adopt Q-learning because it is conceptually simpler, and has been found empirically to converge faster in many cases [4]. Although many existing Q-learning approaches applied in control problems aim at tasks of continuous domains, they select the optimal action and derive the control output directly based on finite discrete actions and estimated $Q^*(x,a)$. This may cause the following problems.

1) How to discretize the action space: When a coarse discretization is used, the action is not smooth and the resulting performance is poor. When a fine discretization is used, the number of state-action pairs becomes large, which results in large memory storage and slow learning procedures.
2) The optimal action value is based on the current states x_t and the approximated value $Q(x_t,a)$. This implies that an error in any of these approximations may be incorporated into the action value. If this is the case, the action value a' and the consequent x'_{t+1} may be slightly different from the optimal a and the consequent x_{t+1}. This error may quickly accumulate and produce a different action policy [11], [12] due to different states x'_{t+1} and x_{t+1}.
3) Reinforcement-learning systems often perform poorly at the early stage of learning due to their more or less

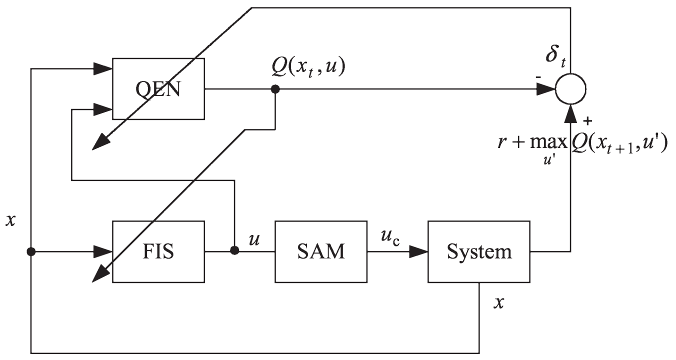


Fig. 1. Architecture of the proposed controller.

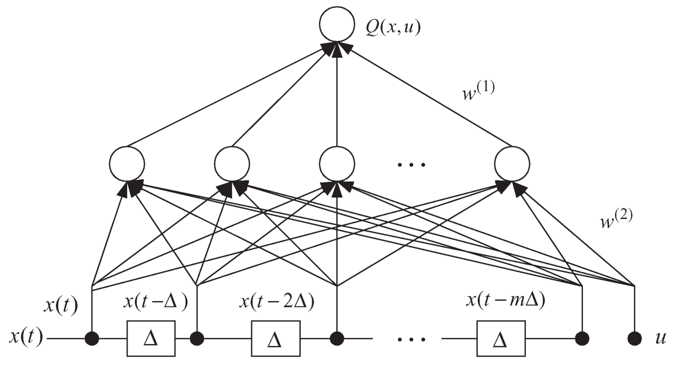


Fig. 2. Architecture of the QEN.

random action. The performance gradually improves as more experience of the world is acquired [11]. The above problem is especially serious in domains where the reward function is largely uniform and dealing with continuous systems with small sample time, since different actions have similar effects.

Assumptions regarding some candidate discrete actions are relaxed in order to eliminate the requirement of discretizing the action space and hence, the first problem alleviated. In order to solve the second problem, the action is not obtained directly based on approximated values of $Q(x, a)$ and candidate discrete actions. Instead, we employ an FIS to provide the control output. The information of the estimated $Q(x, a)$ is only used to tune the parameters of the FIS. A normal FIS can guarantee adequate performance, restrict the control action in a special region, and can be thought of as an initial knowledge. The initial knowledge bootstrapped into the value-function approximation allows the agent to learn more effectively, and helps reduce the time spent acting randomly. Therefore, the third problem is also avoided to some extent.

As a result, our proposed algorithm required less *a priori* knowledge because only an evaluative signal is required for reinforcement learning, and is more applicable to real systems because we combine reinforcement learning with an FIS.

III. The Architecture of the Proposed Controller

As we mentioned before, we use a reinforcement signal to tune the parameters of the fuzzy controller, which provides the control output. So our proposed controller has two main responsibilities: one is to estimate the optimal-action value function $Q^*(x, a)$, and the other is to get the control output based on the estimated-action value function $Q(x, a)$. We construct the controller with two parts—neural network and FIS—to deal with the above two different tasks, respectively.

The overall architecture of this controller is shown in Fig. 1, and is explained in detail in the following parts of this section.

A. Neural Network for Estimating the Optimal-Action Value Function

The $Q^*(x, a)$ estimator network (QEN) plays the role of approximating or predicting the optimal-action value function $Q^*(x, a)$ associated with different input states and control output.

As the approximator of the value function for reinforcement learning, different types of neural networks have been proposed for different tasks and different objectives, such as the adaptive resonance theory (ART) network [15], [16], cerebellar model articulation controller (CMAC) [17], [18], standard multilayer feedforward neural network [19], [20], competitive network [10], radial basis function (RBF) network [21], and neurofuzzy network [22], etc.

In practical environments, many signals are related with time, such as speech signals, measured waveform data, and control signals, etc. In order to deal with the above inputs, the standard multilayer feedforward network may be not appropriate because it is a static network. We adopt a time-delay neural network (TDNN) as the QEN. The TDNN is a back-propagation neural network that uses a time-delayed sequence as its input vector. This allows it to deal with input data that are presented over time.

We use the TDNN to estimate and generalize the optimal-action value function due to the good approximation property of neural networks. The estimation is based on the prediction TD error δ_t that is defined in Section II. The architecture of the QEN is shown in Fig. 2.

Since we want to estimate $Q^*(x, a)$ using the QEN, the input of the TDNN includes system states and control output u, which is the same as the action a mentioned before, and the output of the network is $Q(x, u)$. Here, we turn the temporal sequence of system states $x(t)$ into a spatial input pattern $x(t)$, $x(t - \Delta)$, $x(t - 2\Delta), \ldots, x(t - m\Delta)$. Δ is the sample time, and m is the maximum delay unit to consider.

In our proposed TDNN, $Q(x, u)$ will be the form of

$$\begin{aligned} Q(x,u) &= f(O_1) \\ O_1 &= \sum_{i=1}^{N_\mathrm{h}} w_i^{(1)} p_i \\ p_i &= f(O_{2i}) \\ O_{2i} &= \sum_{j=1}^{N_\mathrm{i}} w_{ij}^{(2)} x_i \end{aligned}$$

where

O_1	the summed input of the output node;
$w_i^{(1)}$	the weight between the ith hidden node and the output node;
p_i	the output of the hidden node, the number of total hidden nodes is N_h;
O_{2i}	the summed input of the ith hidden node;
$w_{ij}^{(2)}$	the weight between the jth input node and ith hidden node;
x_i	including $x(t), x(t-\Delta), x(t-2\Delta), \ldots, x(t-m\Delta)$, and u, and the number of total inputs is N_i, and the $N_\mathrm{i}\mathrm{th}$ input is u;
f	the activation function of the node.

Here, we adopt a sigmoid function as the activation function of the node, i.e., $f(x) = 1/[1+\exp(-x)]$.

The QEN is used to guide the fuzzy controller to tune parameters so that the fuzzy controller will achieve better performance.

B. FIS for Producing the Control Output

Given the current state of the system and the action value function $Q(x,a)$ estimated from the QEN, we may obtain the optimal action u^* based on the candidate discrete action set U and Q-learning algorithm $u^* = \arg_{u\in U} \max Q(x,u)$. Indeed, this is the basic idea of most existing algorithms that adopt Q-learning. But the candidate discrete action set U may not be known to us and the performance of the controller is affected by the selected finite candidate actions accordingly.

The information about the candidate discrete action set U is not required. The FIS, which will be explained in the remainder of the section, generates the control output u that maximizes the action value function $Q(x,u)$ produced by the QEN incrementally. The final output provided by FIS maximizes $Q(x,u)$ with respect to all possible u, instead of the finite candidate discrete action set U.

We propose a FIS as the controller for the following reasons.

1) Compared with obtaining control output directly based on estimated $Q(x,u)$, an FIS is robust with respect to the parameters that are related to the action value function $Q(x,u)$. This means an FIS can achieve acceptable performance even in the early stage of learning, in which the approximation error of $Q(x,u)$ is large.
2) Owing to the fact that reinforcement learning may spend a huge amount of time taking exploratory actions and learning nothing if it cannot search the optimal region, searching the whole space may be time consuming. An FIS, which can incorporate some initial knowledge, may guide the agent to some "good" space, allow it to learn more effectively, and expedite the learning process. There are several references to explain how to design the controller by using the initial knowledge; interested readers may refer to [2].

In this section, we present the FIS architecture. Consider a multiple-input single-output (MISO) fuzzy controller that performs a mapping from a state vector $\underline{x} = (x_1, x_2, \ldots, x_n)^\mathrm{T} \in \Re^n$ to a control input $u \in \Re$. Using the Takagi–Sugeno model, the IF–THEN rules of the fuzzy controller may be expressed as

$$R_l:\ \text{IF } x_1 \text{ is } F_1^l, \ldots, \text{and } x_n \text{ is } F_n^l$$
$$\text{THEN } u = K_0^l + K_1^l x_1 + K_2^l x_2 + \cdots + K_n^l x_n$$

where F_i^l is the label of the fuzzy set in x_i, for $l = 1, 2, \ldots, M$. $K_0^l, K_1^l, K_2^l \ldots$, and K_n^l are the constant coefficients of the consequent part of the fuzzy rule.

We use product inference for the fuzzy implication and t norm, singleton fuzzifier, and center-average defuzzifier; consequently, the final output value is

$$u(\underline{x}) = \frac{\sum_{l=1}^{M}\left(\left(\prod_{i=1}^{n}\mu^{F_i^l}(x_i)\right)\cdot\left(\sum_{j=0}^{n}K_j^l x_j\right)\right)}{\sum_{l=1}^{M}\left(\prod_{i=1}^{n}\mu^{F_i^l}(x_i)\right)} \tag{7}$$

where $\mu^{F_i^l}$ is the membership degree of the fuzzy set F_i^l, $x_0 = 1$.

We adopt a Gaussian function as the membership function of the fuzzy system because of its excellent approximation properties [8], i.e.

$$\mu^{F_i^l}(x_i) = \exp\left(-\left(\frac{x_i - c_i^l}{\sigma_i^l}\right)^2\right) \tag{8}$$

for $i = 1, 2, \ldots, n$ and $l = 1, 2, \ldots, M$.

C. Stochastic Action Modifier

From Section II, we know that $Q_t(x,a)$ converges to $Q^*(x,a)$ with a probability of 1 if all actions continue to be tried from all states. But how can we guarantee all actions to be tried?

In order to solve the above problem, we implement the exploration strategy for the control output u recommended by the FIS. However, some existing exploration strategies such as the Boltzmann exploration and ε-greedy policy mainly deal with discrete actions.

We add a stochastic action modifier (SAM) after the FIS and before the system input. The SAM generates the control command u_c, which is a Gaussian random variable with mean u recommended by the FIS and standard deviation σ_u, and σ_u satisfies the condition that it will converge to zero gradually, i.e., $u_\mathrm{c} = u + \sigma_u(t)\cdot n(t)$, and $\lim_{t\to\infty}\sigma_u(t) = 0$.

IV. Learning Algorithms

In this section, we develop the learning algorithms for the QEN and the FIS. The learning mechanism of the QEN is the combination of the TD methods and the back-propagation algorithm. The learning mechanism of the FIS is based on a gradient algorithm.

A. Learning Algorithm for QEN

From the previous section on reinforcement learning, we know TD methods learn their estimates, in part, on the basis of other estimates. We can also tune the parameters of our proposed QEN based on generalized policy iteration (GPI). We can achieve the task of approximating the optimal-action value function with the neural network by reducing the TD error δ_t continuously.

$$\delta_t = r_{t+1} + \gamma \max_{u'} Q(x_{t+1}, u') - Q(x_t, u_t). \tag{9}$$

In other words, the objective of the neural network is to minimize the following expression.

$$E = \frac{1}{2}\delta_t^2.$$

The weight-update rule for the neural-network-based gradient-descent method is given by

$$w(t+1) = w(t) - \eta \frac{\partial E}{\partial w} \tag{10}$$

$$\frac{\partial E}{\partial w} = \delta_t \frac{\partial \delta_t}{\partial w} = -\delta_t \frac{\partial Q(x_t, u_t)}{\partial w}. \tag{11}$$

Combining the above two equations, we can obtain

$$w(t+1) = w(t) + \eta \delta_t \frac{\partial Q(x_t, u_t)}{\partial w} \tag{12}$$

for $w = w_i^{(1)}$ and $w = w_{ij}^{(2)}$, respectively.

We can obtain $[\partial Q(x_t, u_t)]/\partial w$ based on the chain rules for $w_i^{(1)}$ and $w_{ij}^{(2)}$, respectively.

$$\begin{aligned}\frac{\partial Q(x_t,u_t)}{\partial w_i^{(1)}} &= \frac{\partial Q(x_t,u_t)}{\partial O_1}\frac{\partial O_1}{\partial w_i^{(1)}} = f'(O_1)p_i \\ &= p_i Q(x_t,u_t)\left[1-Q(x_t,u_t)\right] \\ &\quad (\text{for } i=1,\ldots,N_\mathrm{h})\end{aligned} \tag{13}$$

$$\begin{aligned}\frac{\partial Q(x_t,u_t)}{\partial w_{ij}^{(2)}} &= \frac{\partial Q(x_t,u_t)}{\partial O_1}\frac{\partial O_1}{\partial p_i}\frac{\partial p_i}{\partial O_{2i}}\frac{\partial O_{2i}}{\partial w_{ij}^{(2)}} \\ &= f'(O_1)w_1^{(1)} f'(O_{2i})x_j \\ &= w_i^{(1)} x_j Q(x_t,u_t)\left[1-Q(x_t,u_t)\right] p_i[1-p_i] \\ &\quad (\text{for } i=1,\ldots,\ N_\mathrm{h}, j=1,\ldots,N_\mathrm{i})\end{aligned} \tag{14}$$

where the fact that $f'(O) = f(O)[1-f(O)]$ is used for the sigmoid function.

And we can also obtain $[\partial Q(x_t, u_t)]/\partial u$

$$\begin{aligned}\frac{\partial Q(x_t,u_t)}{\partial u} &= \frac{\partial Q(x_t,u_t)}{\partial O_1}\sum_{i=1}^{N_\mathrm{h}}\left(\frac{\partial O_1}{\partial p_i}\frac{\partial p_i}{\partial O_{2i}}\frac{\partial O_{2i}}{u}\right) \\ &= f'(O_1)\sum_{i=1}^{N_\mathrm{h}}\left(w_i^{(1)} f'(O_{2i}) w_{i,\,N_\mathrm{i}}^{(2)}\right) \\ &= Q(x_t,u_t)\left[1-Q(x_t,u_t)\right] \\ &\quad \times \sum_{i=j}^{N_\mathrm{h}}\left(w_i^{(1)} w_{i,\,N_\mathrm{i}}^{(2)} p_i[1-p_i]\right)\end{aligned} \tag{15}$$

where it is noticed that control output of the FIS is the $N_\mathrm{i}\mathrm{th}$ input, i.e., $x_{N_\mathrm{i}} = u$.

In (14) and (15) above, $w_i^{(1)} = 0$ if the $i\mathrm{th}$ hidden node is not connected to the output node, and $w_{i,j}^{(2)} = 0$ if the $j\mathrm{th}$ input node is not connected to the $i\mathrm{th}$ hidden node.

Eligibility traces record the past and current gradients, and adopting eligibility traces could speed up the learning process. The eligibility traces for the QEN is defined as

$$\vec{e}_t = \gamma\lambda\vec{e}_{t-1} + \frac{\partial Q(x_t,u_t)}{\partial w} \tag{16}$$

with $\vec{e}_0 = \vec{0}$.

Then, the parameters of the learning algorithm for the QEN is derived as follows

$$w(t+1) = w(t) + \eta\delta_t\vec{e}_t. \tag{17}$$

B. Learning Algorithm for the FIS

Now, we consider how to improve the control policy using the associated value function. In other words, we consider how to tune the parameters of the fuzzy controller based on the approximated $Q(x,u)$ obtained from the previous section.

For the discrete finite actions, we may obtain the optimal-control action based on finite comparisons of the value function we have approximated. However, this approach is obviously infeasible for continuous actions.

For continuous-time systems without discrete states and actions, we may generalize the value function ($V(x)$ or $Q(x,u)$) using a neural network based on reinforcement signal δ_t, so we can deal with continuous states. But how to deal with continuous actions is not a trivial task. Gullapalli proposed the stochastic real-valued (SRV) unit algorithm, in which the parameters of the action network are updated by gradient estimation [23]. Baird proposed the advantage-updating method, in which both the value function $V(x)$ and the advantage function $A(x,u)$ are updated [24]. It should be noticed here that the above algorithms obtain the continuous actions based on the state value function $V(x)$, not the action value function $Q(x,u)$.

In order to optimize the output of the FIS, we can update the parameters of the FIS to maximize the action value function $Q(x,u)$ with respect to the control output u for the current state. As a result, the learning algorithms of the FIS can be derived using gradient rules

$$\xi(t+1) = \xi(t) + \beta\frac{\partial Q(x_t,u_t)}{\partial \xi} \tag{18}$$

$$\frac{\partial Q(x,u)}{\partial \xi} = \frac{\partial Q(x,u)}{\partial u}\frac{\partial u}{\partial \xi} \tag{19}$$

where ξ are parameters to be tuned in fuzzy systems such as K_j^l, c_i^l and σ_i^l. We have obtained $[\partial Q(x_t,u_t)]/\partial u$ already in the previous section [see (15)]. We will only need to deduce $\partial u/\partial \xi$.

If we let

$$z^l = \prod_{i=1}^{n} \exp\left(-\left(\frac{x_i - c_i^l}{\sigma_i^l}\right)^2\right)$$
$$\overline{y}^l = K_0^l + K_1^l x_1 + \cdots + K_n^l x_n$$
$$a = \sum_{i=1}^{M}(\overline{y}^l z^l), b = \sum_{l=1}^{M} z^l, u = \frac{a}{b}. \tag{20}$$

Then we can calculate $\partial u / \partial \xi$ by the following equations:

$$\frac{\partial u}{\partial K_j^l} = \frac{z^l}{b} x_j \tag{21}$$
$$\frac{\partial u}{\partial c_i^l} = \frac{\partial u}{\partial z^l} \cdot \frac{\partial z^l}{\partial c_i^l} = \frac{\overline{y}^l - u}{b} z^l \frac{2\left(x_i - c_i^l\right)}{\left(\sigma_i^l\right)^2} \tag{22}$$
$$\frac{\partial u}{\partial \sigma_i^l} = \frac{\partial u}{\partial z^l} \cdot \frac{\partial z^l}{\partial \sigma_i^l} = \frac{\overline{y}^l - u}{b} z^l \frac{2\left(x_i - c_i^l\right)^2}{\left(\sigma_i^l\right)^3}. \tag{23}$$

C. Implementation Procedures

We present the detailed implementation procedures as follows.

1) Initialize $Q(x_t, u_t)$, the parameters $w^{(1)}$, $w^{(2)}$ of the TDNN, and the parameters ξ of the FIS, and assume the previous states of the system are all zero (some previous states should be reserved for the input of the TDNN).
2) Obtain the new control output u_{t+1} based on (7) and input of the FIS.
3) Before it is fed to the actual system, u is processed by the SAM according to $u_c = u + \sigma_u(t) \cdot n(t)$.
4) The SAM provides u_c, which acts as the control value of the system.
5) Based on our requirements for the system, we evaluate the performance of the controller as r, which is only the "evaluative signal" instead of the "teacher" signal. And we also obtain the states of the system.
6) Obtain the approximated $Q(x_{t+1}, u_{t+1})$ from the TDNN based on the current control action, the current states, and some previous states.
7) From r, $Q(x_t, u_t)$, and $Q(x_{t+1}, u_{t+1})$, we can calculate the TD error δ_t based on (9). Here, we assume $Q(x_{t+1}, u_{t+1}) \approx \max_{u'} Q(x_{t+1}, u')$ because u_{t+1} is obtained from the FIS, which continuously maximizes $Q(x, u)$ with respect to the control output u.
8) Based on δ_t obtained from step 7), we can update the parameters of the QEN (TDNN) according to (13), (14), (16), and (17).
9) Tune the parameters of the FIS based on (18)–(23).
10) Substitute $Q(x_t, u_t)$ with $Q(x_{t+1}, u_{t+1})$.
11) If the parameters of the QEN and FIS are not changed any more or after predefined iterations, the learning procedure is terminated, otherwise, return to step 2) after a fixed sampling time Δ.

V. Simulation Results

Advanced vehicle control has drawn more and more attention and worldwide research because autonomous driving cannot only free humans from tedious driving tasks, it can also improve safety due to the fast response of the electronic device. A few automated vehicle-control systems have been implemented in real vehicles, for example, autonomous navigation research at Carnegie Mellon University [26], the PATH program at Berkeley [27], Nissan's PVS project [28], and some active research in Korea [29]. The research of autonomous vehicles may be divided into sensors, signal processing, control computation, vehicle dynamics, etc. Most of the current research may concentrate on the new technologies of sensors and image processing of the road scenery, but not so much on the vehicle control itself.

The proposed fuzzy reinforcement-learning controller has been applied for the vehicle longitudinal-control problem, and some simulation results are presented as the first stage of the autonomous vehicle research.

The vehicle longitudinal-control problem addresses the control of a vehicle to maintain a safe distance between the controlled car and the preceding cars. In our simulation, there are four input-state variables available in this system: the position of the preceding car x_l, the velocity of the preceding car v_l, the position of the following car x_f, and the velocity of the following car v_f. For longitudinal control, the control action for the vehicle may be throttle angle command and brake torque command, but also can be considered simply as one control action u (or control effort, engine input), i.e., positive value for throttle angle command and negative value for torque command.

The model and corresponding parameters of the vehicle longitudinal dynamics are adopted from [9], and they are shown as follows:

$$\ddot{x} = \frac{F - c\dot{x}^2 - d}{M} \tag{24}$$
$$\dot{F} = \frac{1}{\tau}(-F + u) \tag{25}$$

where, in the first equation, x, F, c, d, and M are the position, the engine traction force, effective aerodynamic drag coefficient, rolling resistance friction, and effective inertia, respectively. If we consider the engine dynamics, we have the additional (25), where the engine traction force F can be modeled as a first-order system, and u is the control input. $c = 0.44\ \mathrm{kg/m}$, $d = 352\ \mathrm{kg \cdot m/s^2}$, $\tau = 0.2\ \mathrm{s}$, and $M = 1500\ \mathrm{kg}$.

In designing the controller, the dynamic model of the car is assumed to be unknown to the controller. Our proposed controller adopts reinforcement learning to tune the fuzzy controller, so it is a model-free paradigm that is capable of learning the optimal strategy through trial-and-error interactions with a dynamic environment. Unlike some vehicle controllers with neural networks or fuzzy systems [30]–[33], it also has the advantage that the collected input–output vector pairs for different situations (such as different speed and different road situations) are not required.

Since the objective of the longitudinal controller is to maintain a safe distance between the preceding and following cars,

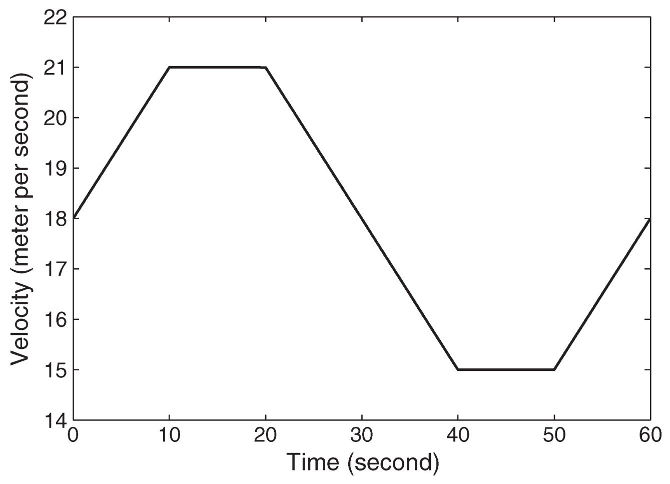


Fig. 3. Velocity profile of the leading car.

we consider the spacing deviation $\Delta x = x_{\mathrm{l}} - x_{\mathrm{f}}$ and relative speed $v_{\mathrm{r}} = v_{\mathrm{l}} - v_{\mathrm{f}}$ as the two input variables of the FIS. Then the fuzzy rules of the adaptive fuzzy controller are considered as follows.

$$R_1: \text{ IF } \Delta x \text{ is } F_1^l \text{ and } v_{\mathrm{r}} \text{ is } F_2^l$$
$$\text{THEN } u = K_0^l + K_1^l \Delta x + K_2^l v_{\mathrm{r}}.$$

We adopt nine fuzzy rules to construct the controller, and initially we define three fuzzy sets over the interval [-5, 5] for Δx and three fuzzy sets over the interval [-0.5, 0.5] for v_{r}.

The input vector to the QEN consisted of seven components or nodes: spacing deviation Δx, relative speed v_{r}, and the control input u. The number of input nodes relates to the time dependence of the drivers' responses to spacing deviation and relative speed because we adopt a TDNN as the QEN. In our simulation, we consider the last three states, i.e., $m = 3$ in the TDNN. The output to the QEN is $Q(x, u)$ corresponding to the action value function of reinforcement learning. The topology of the QEN is considered to be a three-layer structure having 7–21–1 nodes.

In this simulation, the velocity profile of the leading car is shown in Fig. 3.

The next step is to define the reinforcement signal r. We evaluate the performance of the controller by the maximum absolute value of the spacing deviation $|\Delta x|$, because the objective of the longitudinal controller is to maintain a safe distance and make the spacing deviation as small as possible. We assume the larger spacing $|\Delta x|$, the poorer the performance of the controller. In our simulations, we assume the maximum allowed spacing deviation is 6 m. Therefore, failure is defined as the spacing deviation between the preceding car and the following car as larger than 6 m. The reinforcement signal r may be defined as follows:

$$r(t) = \begin{cases} -\frac{1}{6}|\Delta x|, & |\Delta x| \le 6 \\ -1, & |\Delta x| > 6, \text{fail}. \end{cases} \tag{26}$$

In our simulation, the fuzzy controller based on reinforcement learning was tested for 200 trials (periods) of the velocity profile of the preceding car. The controller learns the experience and updates the fuzzy parameters continuously based on the reinforcement signal. In our learning procedures, the system is reset to the initial states and resumes learning once the "fail" signal occurs. The simulation was conducted to evaluate the effectiveness of our control design. The parameters used in the simulation are summarized in Table I with the proper notations defined in the following.

TABLE I
SUMMARY OF PARAMETERS USED IN THE SIMULATION

Parameters	γ	λ	$\eta(0)$	$\eta(f)$
Value	0.95	0.9	0.25	0.005

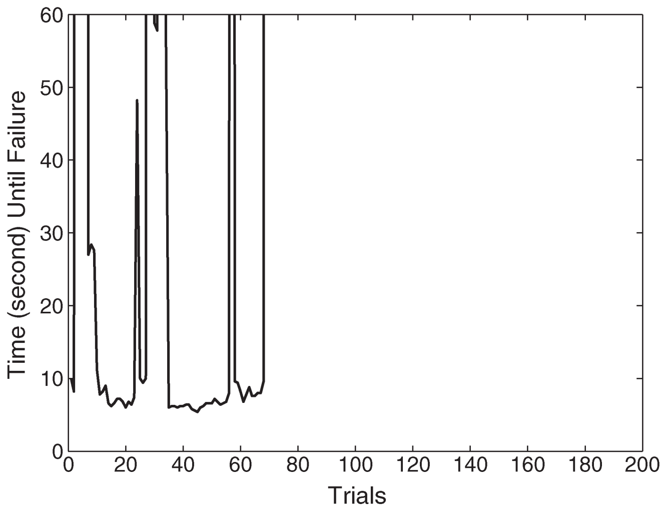


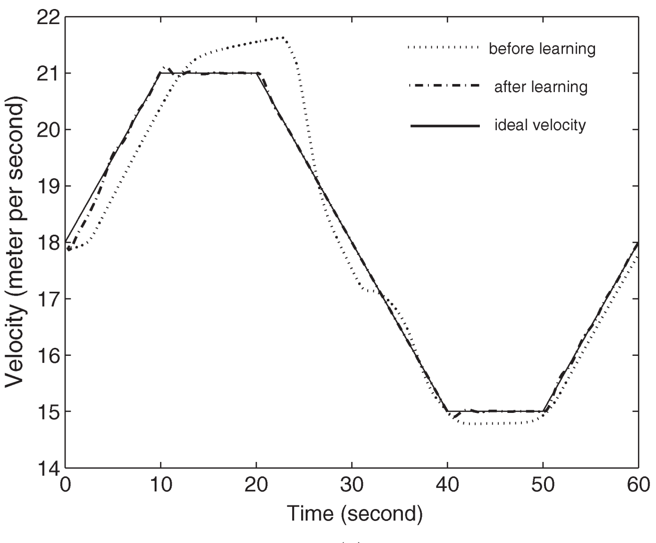
Fig. 4. Performance of the proposed controller on the vehicle longitudinal-control problem.

γ	Discount factor of the TD error δ_t.
λ	Eligibility rate.
$\eta(0)$	Initial learning rate of the QEN.
$\eta(t)$	Learning rate of the QEN at time t, which is decreased with t until it reaches 0.005 and it stays at $\eta(f) = 0.005$.

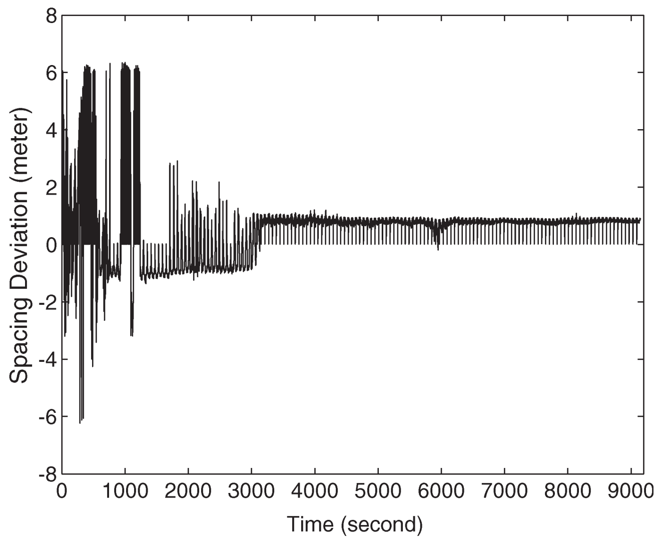
In Fig. 4, the x-axis stands for all the trials simulated, and the y-axis stands for the time until failure for each trial. Here, failure means the controller performance cannot meet our requirements [we defined it in (26)]. Since each trial lasts 80 s, no point for the trial means the controller performance is satisfactory. We can see from Fig. 4 that failures occurred frequently at the beginning of the learning. This means the performance of the initially proposed controller is poor. In other words, the parameters of the controller is not optimal and failures often occurred at time near 10 s because the acceleration of the preceding vehicle is changed suddenly at this time. However, the controller was able to successfully drive the vehicle with the spacing deviation less than 6 m all the times after 68 trials.

We can illustrate the effectiveness of our learning approach by comparing the performance of the controller before learning takes place and after learning takes place. We can see the simulation results clearly from Fig. 5.

In Fig. 5(a), the dotted line is the velocity response of the fuzzy controller before learning takes place, the dash–dot line



(a)



(b)

Fig. 5. Simulation results of the proposed controller. (a) Velocity response of the controller before and after learning takes place; (b) evolution of spacing deviation.

is the velocity response of the fuzzy controller after learning takes place, and the solid line is the velocity of the preceding car. From the simulation results in Fig. 5(a) and (b), we can see that: 1) Not only the spacing deviation is decreased gradually, but also the velocity response is better accordingly. This is a basic characteristic of reinforcement learning, which can continuously improve its performance without a teacher, purely based on trial and error. 2) The velocity response of the controlled car after learning is not identical to the leading velocity. This is due to the sudden velocity change of the preceding car (we can see the deviation is especially obvious for these unexpected situations) and the limit of the number of fuzzy rules. 3) During the learning procedure, the maximum spacing deviation may be increased, this is due to the exploration strategy (SAM) we have implemented. However, the maximum spacing is decreased as a whole. 4) The final spacing deviation is not zero because we only take a few fuzzy rules.

It is worth noting here that the proposed method requires no model information and training data and can achieve better performance with a few fuzzy rules (only nine rules in our simulations).

VI. Conclusion

This paper presents the controller architecture comprised of a QEN and an FIS for solving continuous-space reinforcement-learning problems. The QEN is used to estimate the optimal-action value function, and the FIS is used to get the control output based on the estimated-action value function provided by the QEN. With the proposed architecture, the parameters of the learning algorithms for the QEN and the FIS are developed based on techniques of TD and gradient-descent algorithm. Finally, the simulation studies of the longitudinal control of the vehicle demonstrated the validity and performance of the proposed learning algorithms.

There are still a lot of problems for future consideration. To deal with continuous variables, we adopt a QEN as an approximator to obtain the action value function $Q(x, a)$. However, it has been demonstrated that simply replacing the discrete lookup tables with function approximators may be not robust and cause learning to fail, even in benign cases [11], [12], [36]. So how to generalize carefully from the previously experienced states to ones that have never been met is a future direction for us to consider.

From the simulation, we know that the performance and the learning speed of our proposed controller are affected by the parameters of the controller, such as the standard deviation of the SAM and the learning rate. Currently, we choose these parameters based on a heuristic approach. How to find a systematic approach to select these parameters is important and also our future work.

References

[1] J. E. Naranjo, C. Gonzalez, J. Reviejo, R. Garcia, and T. De Pedro, "Adaptive fuzzy control for inter-vehicle gap keeping," *IEEE Trans. Intell. Transp. Syst.*, vol. 4, no. 3, pp. 132–142, Sep. 2003.
[2] L.-X. Wang, *A Course in Fuzzy System and Control*. Upper Saddle River, NJ: Prentice-Hall, 1997.
[3] ——, "Stable adaptive fuzzy control of nonlinear systems," *IEEE Trans. Fuzzy Syst.*, vol. 1, no. 2, pp. 146–155, May 1993.
[4] R. S. Sutton, A. G. Barto, and R. J. Williams, "Reinforcement learning is direct adaptive optimal control," *IEEE Control Syst. Mag.*, vol. 12, no. 2, pp. 19–22, Apr. 1992.
[5] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.
[6] C.-K. Chiang, H.-Y. Chung, and J.-J. Lin, "A self-learning fuzzy logic controller using genetic algorithms with reinforcements," *IEEE Trans. Fuzzy Syst.*, vol. 5, no. 3, pp. 460–467, Jun. 1997.
[7] C. J. C. H. Watkins, "Learning with delayed rewards," Ph.D. dissertation, Psychology Dept., Cambridge Univ., U.K., 1989.
[8] B. Liu and J. Si, "The best approximation to C^2 function and its error bounds using regular-center Gaussian networks," *IEEE Trans. Neural Netw.*, vol. 5, no. 5, pp. 845–847, Sep. 1994.
[9] D. Swaroop, J. K. Hedrick, and S. B. Choi, "Direct adaptive longitudinal control of vehicle platoons," *IEEE Trans. Veh. Technol.*, vol. 50, no. 1, pp. 150–161, Jan. 2001.
[10] X. W. Yan, Z. D. Deng, and Z. Q. Sun, "Competitive Takagi–Sugeno fuzzy reinforcement learning," in *Proc. IEEE Int. Conf. Control Applications*, San Diego, CA, 2001, pp. 878–883.
[11] W. D. Smart and L. P. Kaelbling, "Practical reinforcement learning in continuous spaces," in *Proc. 17th Int. Conf. Machine Learning*, Stanford, CA, 2000, pp. 903–910.

[12] J. A. Boyan and A. W. Moore, "Generalization in reinforcement learning: Safely approximating the value function," in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. S. Touretzky, and T. K. Leen, Eds. Cambridge, MA: MIT Press, 1995.
[13] G. Tesauro, "TD-Gammon, a self teaching backgammon program, achieves master-level play," *Neural Comput.*, vol. 6, no. 2, pp. 215–219, 1994.
[14] A. G. Barto and M. I. Jordan, "Gradient following without backpropagation in layered networks," in *Proc. IEEE 1st Annu. Conf. Neural Networks*, San Diego, CA, 1987, pp. II629–II636.
[15] C.-J. Lin and C.-T. Lin, "Reinforcement learning for an ART-based fuzzy adaptive learning control network," *IEEE Trans. Neural Netw.*, vol. 7, no. 3, pp. 709–731, May 1996.
[16] C.-T. Lin and I.-F. Chung, "A reinforcement neuro-fuzzy combiner for multiobjective control," *IEEE Trans. Syst., Man, Cybern. B*, vol. 29, no. 6, pp. 726–744, Dec. 1999.
[17] S.-Y. Oh, J.-H. Lee, and D.-H. Choi, "A new reinforcement learning vehicle control architecture for vision-based road following," *IEEE Trans. Veh. Technol.*, vol. 49, no. 3, pp. 997–1005, May 2000.
[18] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, Psychology Dept., Cambridge Univ., U.K., 1989.
[19] C.-T. Lin and C. S. G. Lee, "Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems," *IEEE Trans. Fuzzy Syst.*, vol. 2, no. 1, pp. 46–63, Feb. 1994.
[20] J. Si and Y.-T. Wang, "Online learning control by association and reinforcement," *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 264–276, Mar. 2001.
[21] Y. Koike and K. Doya, "Multiple state estimation reinforcement learning for driving model: Driver model of automobile," in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics (SMC)*, Tokyo, Japan, 1999, vol. 5, pp. 504–509.
[22] L. Jouffe, "Fuzzy inference system learning by reinforcement methods," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 28, no. 3, pp. 338–355, Aug. 1998.
[23] V. Gullapalli, "A stochastic reinforcement learning algorithm for learning real-valued functions," *Neural Netw.*, vol. 3, no. 6, pp. 671–692, 1990.
[24] L. C. Baird, III, "Reinforcement learning in continuous time: Advantage updating," in *IEEE World Congr. Computational Intelligence., IEEE Int. Conf. Neural Networks*, Orlando, FL, 1994, vol. 4, pp. 2448–2453.
[25] E. Zalama, J. Gomez, M. Paul, and J. R. Peran, "Adaptive behavior navigation of a mobile robot," *IEEE Trans. Syst., Man, Cybern. A*, vol. 32, no. 1, pp. 160–169, Jan. 2002.
[26] M. H. Hebert, C. E. Thorpe, and A. Stentz, *Intelligent Unmanned Ground Vehicles: Autonomous Navigation Research at Carnegie Mellon*. Norwell, MA: Kluwer, 1997.
[27] S. Shladover *et al.*, "Automated vehicle control developments in the PATH program," *IEEE Trans. Veh. Technol.*, vol. 40, no. 1, pp. 114–130, Feb. 1991.
[28] M. Taniguchi *et al.*, "The development of autonomously controlled vehicle (PVS)," in *Proc. Vehicle Navigation and Information Syst. Conf.*, Dearborn, MI, 1991, pp. 1137–1141.
[29] D. H. Choi, S. Y. Oh, and K. Kim, "Connectionist–nonconnectionist fusion architecture for high speed road following," *Neural Parallel Sci. Comput.*, vol. 4, no. 3, pp. 367–386, Sep. 1996.
[30] H. M. Kim, J. Dickerson, and B. Kosko, "Fuzzy throttle and brake control for platoons of smart cars," *Fuzzy Sets Syst.*, vol. 84, no. 23, pp. 209–234, Dec. 1996.
[31] S. Huang and W. Ren, "Use of neural fuzzy networks with mixed genetic/gradient algorithm in automated vehicle control," *IEEE Trans. Ind. Electron.*, vol. 46, no. 6, pp. 1090–1102, Dec. 1999.
[32] J. Mar and F.-J. Lin, "An ANFIS controller for the car-following collision prevention system," *IEEE Trans. Veh. Technol.*, vol. 50, no. 4, pp. 1106–1113, Jul. 2001.
[33] N. Kehtarnavaz, N. Groswold, K. Miller, and P. Lascoe, "A transportable neural-network approach to autonomous vehicle following," *IEEE Trans. Veh. Technol.*, vol. 47, no. 2, pp. 694–702, May 1998.
[34] I. H. Suh, J. H. Kim, and F. C.-H. Rhee, "Fuzzy Q-learning for autonomous robot systems," in *Proc. Int. Conf. Neural Networks*, Houston, TX, 1997, vol. 3, pp. 1738–1743.
[35] I. H. Suh, J. H. Kim, and S. R. Oh, "Region-based Q-learning for intelligent robot systems," in *IEEE Int. Symp. Computational Intelligence Robotics and Automation (CIRA)*, Monterey, CA, 1997, pp. 172–178.
[36] L. Barid and A. Moore, "Gradient descent for general reinforcement learning," in *Advances in Neural Information Processing Systems 11*. Cambridge, MA: MIT Press, 1999.

Xiaohui Dai received the M.Phil. degree from the Department of Electronic and Information Engineering of The Hong Kong Polytechnic University, Hong Kong, China, in 2004.

He is currently a Research Engineer at the Rockwell Automation Shanghai Research Center, Shanghai, China. His current research interest is on industrial control networks.

Chi-Kwong Li (SM'94) received the degree in instrumentation and control engineering from the University of Westminster, U.K. with first class honors in 1976 and the Masters and Doctors degree from the London University and University of Westminster in 1978 and 1984, respectively.

He is currently an Associate Professor of the Department of Electronic and Information Engineering at The Hong Kong Polytechnic University, Hong Kong, China, where he has been affiliated since 1985. He holds a number of government advisory posts and is active in both academic and consultancy works. He has over 30 years academic and industrial experience in the U.K. and Hong Kong.

He is a Chartered Engineer, a Fellow of the HKIE, the HKAAST; and a Member of the IMechE. He is qualified as a Professional Engineer in Control, Automation and Instrumentation, Electronic, Information and Mechanical Engineering.

A. B. Rad (M'00–SM'02) received the B.Sc. degree in engineering from Abadan Institute of Technology, Abadan, Iran, in 1977, the M.Sc. degree in control engineering from the University of Bradford, Bradford, U.K., in 1986, and the Ph.D. degree in control engineering from the University of Sussex, Brighton, U.K., in 1989.

He is currently a Professor in the Department of Electrical Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong. His current research interests include intelligent control and soft-computing methods, system identification and adaptive control.