A Fuzzy Controller With Supervised Learning Assisted Reinforcement Learning Algorithm for Obstacle Avoidance

Cang Ye, *Member, IEEE,* Nelson H. C. Yung, *Senior Member, IEEE,* and Danwei Wang, *Member, IEEE*

***Abstract*—Fuzzy logic system promises an efficient way for obstacle avoidance. However, it is difficult to maintain the correctness, consistency, and completeness of a fuzzy rule base constructed and tuned by a human expert. Reinforcement learning method is capable of learning the fuzzy rules automatically. However, it incurs heavy learning phase and may result in an insufficiently learned rule base due to the curse of dimensionality. In this paper, we propose a neural fuzzy system with mixed coarse learning and fine learning phases. In the first phase, supervised learning method is used to determine the membership functions for the input and output variables simultaneously. After sufficient training, fine learning is applied which employs reinforcement learning algorithm to fine-tune the membership functions for the output variables. For sufficient learning, a new learning method using modified Sutton and Barto's model is proposed to strengthen the exploration. Through this two-step tuning approach, the mobile robot is able to perform collision-free navigation. To deal with the difficulty in acquiring large amount of training data with high consistency for the supervised learning, we develop a virtual environment (VE) simulator, which is able to provide desktop virtual environment (DVE) and immersive virtual environment (IVE) visualization. Through operating a mobile robot in the virtual environment (DVE/IVE) by a skilled human operator, the training data are readily obtained and used to train the neural fuzzy system.**

***Index Terms*—Fuzzy system, obstacle avoidance, reinforcement learning, supervised learning, virtual environment (VE).**

I. Introduction

THE ultimate goal of mobile robotics research is to endow the robots with high autonomous ability, of which navigation in an unknown environment is achieved by using on-line sensory information. A significant amount of research effort has been devoted to this area in the past decades. Among the proposed methods in the literature, geometry algorithm assumes that the local obstacles are fully recognized via visual sensor [1], [2] or can be learned through on-line acquisition via distance sensor [3], [4]. The former assumption may not be applicable to a real environment, while the latter is time-consuming to explore an unknown environment. On the other hand, potential field method [5], [6] seems more efficient for fast obstacle avoidance as it does not need to know the details of the neighboring obstacles. However, its disadvantages, such as local minimum and unstable motion [7], may limit its practicality.

Since Brooks [8] proposed the behavior control architecture, a similar approach has been adopted [9]–[11] to solve the navigation problem in an unknown environment. Unlike the traditional navigation architecture [12] which decomposes the navigation task using a sense-model-plan-act (SMPA) framework and connects each module serially, the behavior control method decomposes the navigation system into special task-specific behavior modules, e.g., obstacle avoidance, goal seeking, etc., which are connected directly to sensors and actuators and operate in parallel. Therefore, this architecture can act in real-time and has good robustness. As the behavior control architecture tackles the navigation problem in an on-line manner and requires no environment model, it is efficient in dealing with navigation in an unknown environment.

In the behavior control architecture, behavior modules are usually constructed as reactive systems [9]–[11], which map the perceived situations to the correct actions. Fuzzy logic method [13]–[15] is an efficient way of representing this mapping relationship as it is able to represent human expert's knowledge and requires no mathematical model. Furthermore, it is able to describe the input state continuously. For the construction of the behavior modules, obstacle avoidance behavior is the most difficult as it incurs large number of input spaces. It is not easy to define the appropriate fuzzy sets for each input variable and information may be incomplete when human experts express their experience by linguistic rules. In other words, it is intractable to maintain the correctness, consistency, and completeness of the fuzzy rule base compiled and tuned by a human expert for the obstacle avoidance behavior. Therefore, a fuzzy system, which is able to evolve and automatically improve its performance, is highly desired.

A number of learning algorithms, such as evolutionary algorithm [16], reinforcement learning [10], [11], [17]–[21], and supervised learning [22]–[24] have been proposed to construct the fuzzy system automatically. Evolutionary algorithm itself always results in a very long learning process. Reinforcement learning method seems quite promising as it requires no training data. However, it usually leads to a heavy learning phase as the gradient information is not provided explicitly. Due to the large number of the input space for learning obstacle avoidance, the

Manuscript received March 15, 2001; revised January 25, 2002. This work was supported by the research postgraduate studentship from the University of Hong Kong and the research fellowship from Nanyang Technological University. This paper was recommended by Associate Editor D. Y. Lee.

C. Ye is with the Advanced Technologies Laboratory, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: yecang@engin.umich.edu).

N. H. C. Yung is with the Department of Electronic and Electrical Engineering, University of Hong Kong, Hong Kong (e-mail: nyung@eee.hku.hk).

D. Wang is with the School of Electronic and Electrical Engineering, Nanyang Technological University, Nanyang, Singapore (e-mail: edwwang@ntu.edu.sg).

Digital Object Identifier 10.1109/TSMCB.2003.808179

search space becomes too large and the performance evaluation surface becomes too complex to allow efficient learning. Therefore, it is not easy to apply the reinforcement structural and parameter learning methods [25], [26] to learn obstacle avoidance since it is difficult to tell that an incorrect response is due to a mismatch antecedent part or due to an incorrect consequent part [19]. Furthermore, the phenomenon of premature convergence [18], [34] (e.g., trap situation) and ill behavior (e.g., circumnavigate around an obstacle closely and slowly) further undermines the practicality of these methods. On the contrary, supervised learning method has the advantages of fast convergence and is suitable for structure and parameter learning. However, it is very difficult to obtain sufficient training data, which contain no conflict input/output pairs. Insufficient training data may result in an incomplete fuzzy rule base, while the conflicts among the training data may cause incorrect fuzzy rules.

In summary, it is intractable to learn obstacle avoidance behavior by using either reinforcement learning or supervised learning only. However, it is possible to employ supervised learning to reduce the search space of reinforcement learning by pretuning the input and output fuzzy sets first and then apply reinforcement learning to fine-tune the incorrect rules caused by inconsistent training data at supervised learning phase. The benefits of the above approach are as follows.

1) Search domain of the reinforcement learning is greatly reduced by pretuning the rule base. Therefore, the reinforcement learning may be accelerated.
2) As the reinforcement learning starts from a pretuned rule base, insufficient learning or ill behavior may be potentially overcome.
3) Conflicts between rules and incorrect rules induced by supervised learning may be removed by fine-tuning of the reinforcement learning.

Motivated by these observations, we propose a neural fuzzy system with a mixed learning algorithm. It consists of a coarse learning and a fine learning phase. In the coarse phase, supervised learning method is used to determine the input and output fuzzy sets simultaneously. After sufficient training, the membership functions of the input variables are frozen and fine learning phase employing reinforcement learning algorithm is applied to further tune the output fuzzy sets. In order to maintain a relatively high consistency for the training data, we develop a virtual environment (VE) simulator, which is able to provide desktop virtual environment (DVE) and immersive virtual environment (IVE) visualization. Through operating a mobile robot in the DVE/IVE by a skilled human operator, the training data are gradually obtained and used to train the neural fuzzy system.

This paper is organized as follows. Section II introduces the basic concept and framework of the neural fuzzy system and defines the mobile robot model and coordinate systems. Section III presents the mixed learning algorithm where the method to accelerate the supervised learning and the approach to strengthen the exploration are discussed. Section IV describes a VE simulator used for collecting the training data. Section V depicts the simulation results of the proposed learning algorithm and some comparisons with related methods; and Section VI presents the performance analysis of the fuzzy system constructed by the proposed learning algorithm. Finally, conclusions are given in Section VII.

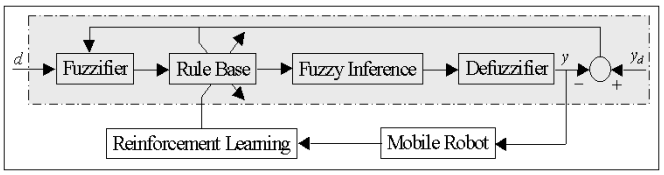

Fig. 1. Diagram of the neural fuzzy system with mixed learning algorithm.

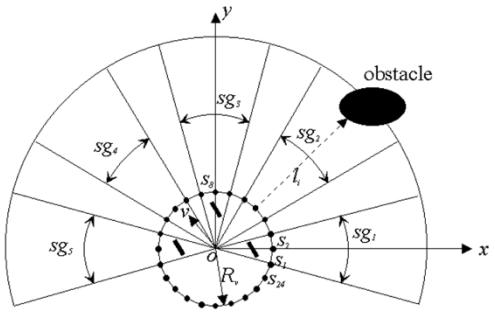

Fig. 2. Mobile robot and sensor arrangement.

II. Neural Fuzzy System

A. General Overview

The proposed neural fuzzy system (see Fig. 1) employs a mixed learning algorithm—supervised learning and reinforcement learning. In the first learning phase, supervised learning method (depicted as gray) is applied. For each input state vector d, the system infers an output y. The difference between the system output y and the desired output y_d is used to train the neural fuzzy system such that the parameters of the input and output fuzzy sets are determined. In the second learning phase, the parameters of the input fuzzy sets are frozen, and reinforcement learning method is employed to further tune the parameters of the output fuzzy sets. For each input state d, the system infers an output y, which is applied to the mobile robot after adding a stochastic perturbation; and the mobile robot moves to a new state. By evaluating the new state, an internal reinforcement signal is generated and is used to fine-tune fuzzy system. In both learning phases, the system performance is improved gradually with the learning proceeds.

B. Mobile Robot Model and Coordinate Systems

As depicted in Fig. 2, we used a cylindrical mobile robot model with a radius of 20 cm. The robot is omnidirectional and there are 24 ultrasonic sensors evenly distributed in a ring. Each sensor, s_i for $i = 1, \ldots, 24$, covers an angular view of 15° and gives the distance to the obstacle l_i in its field of view. To reduce the input dimension, the sensors in the front of the robot are divided into five sensor groups (denoted by sg_i for $i = 1, \ldots, 5$), each of which consists of three neighboring sensors. With this sensor arrangement, the distance d_i measured by the ith sensor group from the center of the mobile robot to the obstacle is expressed as

$$d_i = R_v + \min(l_j \,|\, j = 3i-2, 3i-1, 3i); \quad \text{for } i = 1, \ldots, 5. \tag{1}$$

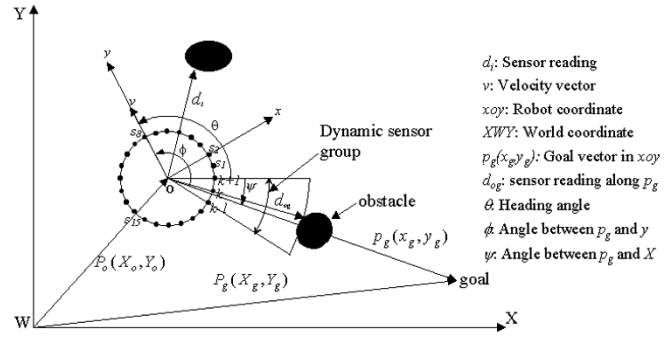

Fig. 3. Diagram of the coordinate systems and control variables.

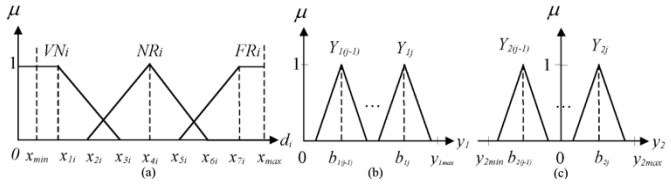

Fig. 4. Membership functions for the input and output variables.

The remaining sensors are used to compose a dynamic sensor group to detect the obstacle distance along the goal direction for behavior fusion [27].

We use two coordinate systems (see Fig. 3); the world coordinate denoted by XWY and the mobile robot coordinate given by xoy where o is the center of the robot and y axis goes through the center of sensor s_8 (the robot will move straight ahead when the wheels are aligned with y axis). The control variables of the robot are the linear velocity v and the change in the heading angle $\Delta\theta$ (referred as steering angle hereafter). In term of behavior control architecture, the purpose of the obstacle avoidance behavior is to determine an action v_a and $\Delta\theta_a$ (denoted by y_1 and y_2, respectively, for simplicity) for each input state $d = (d_1, d_2, \ldots, d_5)$ without considering whether it will cause a deviation from the goal.

C. Construction of the Fuzzy System

Considering the omnidirectional kinematic nature and the symmetry of the robot and the sensor arrangement, each input variables is assigned the same number of fuzzy sets. We use three fuzzy sets in this research to maintain appropriate number of rules. The membership functions of the input and output variables are illustrated in Fig. 4. In Fig. 4(a), the crisp value of each input variable d_i is fuzzified and expressed by the fuzzy sets VN_i, NR_i, and FR_i, referring to very near, near, and far, respectively. d_i is bounded by the minimum value $d_{\min} = R_v + l_{\min}$ and the maximum value $d_{\max} = R_v + l_{\max}$, where $R_v, l_{\min}$ and $l_{\max}$ are the radius of the robot, the minimum and the maximum detectable distance, respectively. The parameters x_{ri} for $r = 1, \ldots, 7$ and $i = 1, \ldots, 5$, are bounded by $x_{\min}$ and $x_{\max}$, where $x_{\min} = d_{\min}$ and $d_{\min} < x_{\max} \leq d_{\max}$. The fuzzy sets VN_i, NR_i, and FR_i are described by

$$\mu_{VN_i}(d_i) = \begin{cases} (x_{3i} - d_i)/(x_{3i} - x_{1i}), & x_{1i} \leq d_i \leq x_{3i} \\ 1, & 0 \leq d_i < x_{1i} \\ 0, & \text{otherwise} \end{cases} \tag{2a}$$

$$\mu_{NR_i}(d_i) = \begin{cases} (d_i - x_{2i})/(x_{4i} - x_{2i}), & x_{2i} \leq d_i \leq x_{4i} \\ (\bar{x}_{6i} - d_i)/(x_{6i} - x_{4i}), & x_{4i} < d_i \leq x_{6i} \\ 0, & \text{otherwise} \end{cases} \tag{2b}$$

$$\mu_{FR_i}(d_i) = \begin{cases} (d_i - x_{5i})/(x_{7i} - x_{5i}), & x_{5i} \leq d_i \leq x_{7i} \\ 1, & d_i > x_{7i} \\ 0, & \text{otherwise.} \end{cases} \tag{2c}$$

The fuzzy rule base consists of 243 rules and it requires 243 fuzzy sets, Y_{1j} for $j = 1, \ldots, 243$, to represent y_1 and 243 fuzzy sets, Y_{2j} for $j = 1, \ldots, 243$, to represent y_2. The fuzzy sets of the output variables y_1 and y_2 take the triangular membership functions, as shown in Fig. 4(b) and (c), respectively, while their center positions, b_{ij} and b_{2j} for $j = 1, \ldots, 243$, are determined by the proposed learning algorithm. $y_{1\max}$ is the upper bound for y_1; while $y_{2\min}$ and $y_{2\max}$ are the lower and upper bound for y_2, respectively. The fuzzy rule is denoted by

$$\begin{aligned} \text{Rule}j\text{: IF } d_1 \text{ is } D_{j1} \text{ AND } \cdots \text{ AND } d_5 \text{ is } D_{j5} \\ \text{THEN } y_1 \text{ is } Y_{1j}, y_2 \text{ is } Y_{2j}; \quad \text{for } j = 1, \ldots, 243 \end{aligned}$$

where D_{ji} is the fuzzy set for d_i in the jth rule, which takes the linguistic value of VN_i, NR_i, or FR_i; and Y_{1j} and Y_{2j} are the fuzzy sets for y_1 and y_2, respectively, in the jth rule. If Larsen's product inference and height defuzzification method is used, the control output of the neural fuzzy system for an input $d' = (d'_1, \ldots, d'_5)$ is given by

$$y_m = \frac{\sum_{j=1}^{243} \mu_j(d') b_{mj}}{\sum_{j=1}^{243} \mu_j(d')} \quad \text{for } m = 1, 2 \tag{3}$$

where $\mu_j(d')$ is the fired strength of the jth rule and is calculated by

$$\mu_j(d') = \mu_{D_{j1}}(d'_1)\mu_{D_{j2}}(d'_2)\mu_{D_{j3}}(d'_3)\mu_{D_{j4}}(d'_4)\mu_{D_{j5}}(d'_5). \tag{4}$$

III. Automatic Rule Generation by the Mixed Learning Algorithm

A. Supervised Learning

We use the error between the actual output y_2 and the desired output y_{2d} to derive the learning algorithm. As the input and output are in different metrics, they must be normalized to $\bar{y}_2$ and $\bar{y}_{2d}$, respectively. The learning process is to minimize the following objective function for the input $d' = (d'_1, \ldots, d'_5)$

$$J = \frac{1}{2}(\bar{y}_2 - \bar{y}_{2d})^2. \tag{5}$$

Using the steepest descent learning algorithm [28], we can derive delta rule (DR) as follows:

$$x_{ri}(k+1) = x_{ri}(k) - \rho\eta \frac{\partial J}{\partial x_{ri}}(k) \tag{6a}$$

$$b_{2j}(k+1) = b_{2j}(k) - \rho\eta \frac{\partial J}{b_{2j}}(k) \tag{6b}$$

where $\eta, 0 < \eta \leq 1$ is the learning rate; k is the number of iterations; and $\rho = (x_{\max} - x_{\min})^2/(y_{\max} - y_{\min})^2$ is a constant induced by the denormalization. $\partial J/\partial x_{ri}(k)$ and $\partial J/\partial b_{2j}(k)$ can be derived by chain rule. We omit it here for simplicity. To

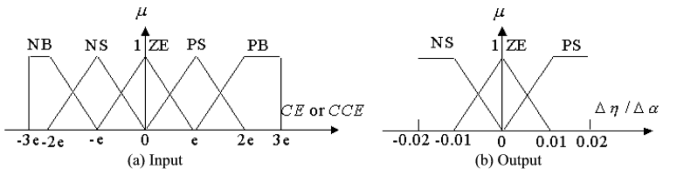

Fig. 5. Membership functions of CE, CCE, $\Delta\eta$ and $\Delta\alpha$: ZE = zero; PS = positive small; PB = positive big; NS = negative small; and NB = negative big.

TABLE I
FUZZY RULES FOR THE ADAPTATION OF η AND α

CCE \ CE	NB	NS	ZE	PS	PB
NB	NS, NS	NS, NS	NS, ZE	NS, ZE	NS, ZE
NS	NS, NS	ZE, ZE	PS, ZE	ZE, ZE	NS, ZE
ZE	ZE, ZE	PS, PS	PS, PS	PS, PS	ZE, ZE
PS	NS, ZE	ZE, ZE	PS, ZE	ZE, ZE	NS, NS
PB	NS, ZE	NS, ZE	NS, ZE	NS, NS	NS, NS
	$\Delta\eta$, $\Delta\alpha$				

increase the learning speed and avoid the problem of instability, a momentum term is applied to (6a) and (6b). This yields the following general delta rule (GDR)

$$x_{ri}(k+1) = x_{ri}(k) + \alpha\Delta x_{ri}(k-1) - \rho\eta\frac{\partial J}{\partial x_{ri}}(k) \quad (7a)$$

$$b_j(k+1) = b_j(k) + \alpha\Delta b_j(k-1) - \rho\eta\frac{\partial J}{b_j}(k). \quad (7b)$$

B. Parameter Adaptation by Fuzzy Control

Let $E = y_2 - y_{2d}$; the change of E is denoted as CE, and the change of CE is denoted as CCE. To increase the convergence rate and prevent oscillations, we may use the following heuristics to adapt η and α from cycle to cycle.

1) If CE is small with no sign changes in several consecutive iterations, η should be increased [29].
2) If CE changes sign in several consecutive iterations, η should be decreased, regardless of the value of CCE [29].
3) IF both CE and CCE are small and have not changed sign for several consecutive iterations, both η and α should be increased [30].

A fuzzy system with two-input (CE and CCE) and two-output ($\Delta\eta$ and $\Delta\alpha$) is adopted to implement the parameter adaptation. The membership functions for the input variables are depicted in Fig. 5(a), where e is determined by the convergence criteria; and the membership functions for the output variables are depicted in Fig. 5(b). The fuzzy rules for adapting the value of η and α are given in Table I with the fuzzy rules for η in the first column followed by the fuzzy rules for α.

Finally, the crisp value of $\Delta\eta$ and $\Delta\alpha$ is determined by the center-of-area defuzzification method and applied to the learning parameters. Then, the next iteration is made using (7a) and (7b). It is called the *general delta rule with fuzzy parameter adaptation (GDRFPA).*

C. Reinforcement Learning

1) Modified Sutton and Barto's Model: After the supervised learning phase, the parameters x_{ri} are frozen, and reinforcement

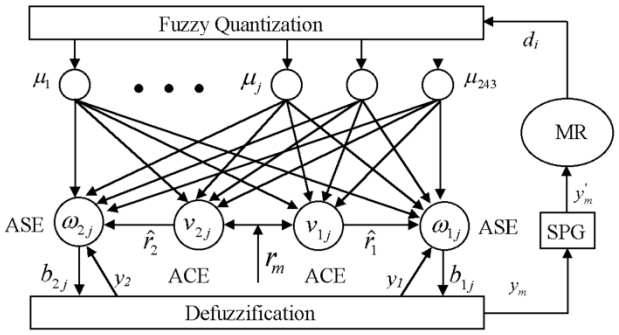

Fig. 6. Neural fuzzy system employing reinforcement learning.

learning is applied to further tune the parameters b_{1j} and b_{2j}. In this paper, we adopt Sutton and Barto's model [17], [18], and modify the network of the learning algorithm, as depicted in Fig. 6.

The mobile robot begins the learning with an initial configuration at time step $t = 0$, where it acquires the environmental information d_i and infers the recommended action $y_m(0)$. Based on the performance evaluation of the system (reinforcement signal) and $y_m(0)$, the stochastic perturbation generator (SPG) (described in the next section) generates an action $y_m(0)$, which is applied to the robot. This moves the robot to a new configuration at time step $t = 1$, and so on, until a collision occurs at the time step $t = k$. The whole process, until a collision occurred, is called a *trial*. For instance, if a trial ends at $t = k$ where a collision occurs, then a reinforcement signal r_m, representing the failure, is fed back to the learning network, and the rules which were used at the previous time steps $k, k-1, k-2, \ldots$, would be changed in order to improve the robot's performance. In Fig. 6, this task is accomplished by an adaptive neuron-like element, which consists of an associative search element (ASE) and an associative critic element (ACE). After the rules are updated, a new trial begins at $t = k+1$. The process is iterated and terminated until no more collision occurs.

Suppose that the current configuration of the robot is $S(t) = (X_0(t), Y_0(t), \theta_0(t))^T$, at which the sensor readings d_i are encoded into $\mu_j(t)$ by (4). In order to obtain the associativity of learning the rules, the trace $\mu_j(t)$ of the fired jth rule is used and is calculated by

$$\bar{\mu}_j(t+1) = \lambda\bar{\mu}_j(t) + (1-\lambda)\mu_j(t) \quad (8)$$

where $\lambda, 0 \le \lambda < 1$ is the trace decay rate. Each ACE receives an external reinforcement signal $r_m(t)$ as a performance feedback, and generates an internal reinforcement signal $\hat{r}_m(t)$, which are fed into the ASE to update its weights. The external reinforcement signal is determined by (9), shown at the bottom of the next page, where ΔT is the time interval between two learning steps and $V_{\max}$ is the upper bound of the robot's velocity. In order to determine the internal reinforcement signal $\hat{r}_m(t)$, two ACEs are used to predict the discounted sum

$$z_m(t) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r_m(t'+1) \quad (10)$$

where $\gamma, 0 < \gamma < 1$, is the discount rate. If the prediction value of $z_m(t)$, denoted by $p_m(t)$, are accurate, from (10) we have

$$p_m(t-1) = r_m(t) + \gamma p_m(t). \tag{11}$$

In practice, the ACEs learn to make the predictions. Thus, the mismatch or time difference error between the two sides of (11) is defined as the internal reinforcement signal and expressed as

$$\hat{r}_m(t) = r_m(t) + \gamma p_m(t) - p_m(t-1). \tag{12}$$

The prediction $p_m(t)$ is implemented as a weighted sum of $\mu_j(t)$ and given by

$$p_m(t) = G\left(\sum_{j=1}^{243} v_{mj}(t)\mu_j(t)\right) \tag{13}$$

where v_{mj}, for $m = 1, 2$ and $j = 1, \ldots, 243$, are the weights of the ACEs and $G(x) = 2/(1 + \exp(\xi x)) - 1$ is a bi-stable function. In order to give correct prediction value, the weights of each ACE are learned through the trace of the fired rule $\bar{\mu}_j(t)$ and its output $\hat{r}_m(t)$. They are updated by

$$v_{mj}(t+1) = v_{mj}(t) + \beta \hat{r}_m(t)\bar{\mu}_j(t) \tag{14}$$

where β is a positive constant determining the rate of change for v_{mj}. Similarly, the weights of the ASE, ω_{mj} for $m = 1, 2$ and $j = 1, \ldots, 243$, are updated by

$$\omega_{mj}(t+1) = \omega_{mj}(t) + \alpha \hat{r}_m(t) e_{mj}(t) \tag{15}$$

where $\alpha, 0 < \alpha \leq 1$ is the learning rate and $e_{mj}(t)$ is the eligibility trace of the jth rule and is updated by

$$e_{mj}(t+1) = \delta e_{mj}(t) + (1-\delta) y'_m(t)\mu_{mj}(t) \tag{16}$$

where $\delta, 0 \leq \delta < 1$ is the decay rate of the eligibility; and $y'_m(t)$ is the actual action applied to the robot which is a Gaussian random variable generated by the SPG. The eligibility trace describes that certain rules have been used and what control actions have been applied. The center positions of the fuzzy sets for the output variables at each time step are determined by

$$b_{mj}(t) = b_m + \frac{\omega_{mj}(t) f_m}{k \max(|\omega_{mj}(t)|) + |\omega_{mj}(t)|} \quad \text{for } j = 1, \ldots, 243 \tag{17}$$

where b_m is the initial center position of the fuzzy sets; f_m is a positive constant that determines the range of b_{mj}; and the positive constant k is used to guarantee the fuzzy sets for the output variable to be within their universe of discourse. The recommended action $y_m(t)$ can be calculated by (3) and the actual action $y'_m(t)$ is determined by the SPG based on $y_m(t)$ and $\hat{r}_m(t)$.

2) Stochastic Perturbation Generator: For learning obstacle avoidance, there is a conflict between 1) the desire to use the rule base already learned; and 2) the desire to further explore the environment so as to make improvement on the rule base. This phenomenon is called *the conflict between exploration and exploitation* [31]. The existing method using Sutton and Barto's model [17]–[21] may result in insufficiently learned rule base as they used pure exploitation. To overcome this drawback and maintain the efficiency of learning, a tradeoff between exploration and exploitation should be achieved.

This objective is achieved by using the SPG. The SPG generates an action $y'_m(t)$, which is a Gaussian random variable with mean $y_m(t)$ and standard deviation $\sigma_m(t)$. $y'_m(t)$ is the actual action applied to the robot, while $\sigma_m(t)$ is a nonnegative function given by

$$\sigma_m(t) = \begin{cases} a_m(e^{-\psi \hat{r}_m(t)} - \tau), & e^{-\psi \hat{r}_m(t)} > \tau \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

where

- ψ search domain scaling factor and is set to 1.0;
- a_m constant;
- $0 \leq \tau \leq 1$ threshold value determining the strength of exploration.

If τ is small, $\sigma_m(t)$ will be large which may slow down the learning, but lead to an extensive exploration. With the specific value of a_m and τ, the perturbation is large when $\hat{r}_m(t)$ is low and small when $\hat{r}_m(t)$ is high. As $\hat{r}_m(t)$ is a performance evaluation of the previous action, the result is that a large random error away from the recommended action results when the previous action performed is bad, but the SPG remains consistent with the fuzzy rules when the previous action is a good one. The learning system converges at specific performance for each set of a_m and τ value. As can be seen from (18), perturbation is allowed when the learning converges. An obstacle avoidance behavior learned using the SPG is able to keep a large clearance from obstacles, as a trajectory closer to an obstacle is more likely to get collision hence not stable. Compared with the existing methods [25], [26], [32], the exploration strength of the SPG is adjustable by τ and we use internal reinforcement in SPG instead of prediction [25], [26].

With the known control action of the current time step, the robot's configuration is updated by

$$S(t+1) = \begin{pmatrix} \theta(t) + y'_2(t) \\ X_o(t) + y'_1(t)\Delta T \cos(\theta(t) + y'_2(t)) \\ Y_o(t) + y'_1(t)\Delta T \sin(\theta(t) + y'_2(t)) \end{pmatrix}. \tag{19}$$

Eventually, if the rules are sufficiently learned in a specific environment, the weights of the ASEs will converge to a set of fixed values. When the learning process is terminated, the learned set of b_{mj} is used as the rule base for the obstacle avoidance module.

IV. Training Pattern Acquisition

The simulation platform used to acquire training data is based on the EXPECTATIONS simulator [33], which is able to

$$r_m(t) = \begin{cases} -1, & \text{if } \min(d_i \,|\, i = 1, 2, \ldots, 5) < d_{\min} + V_{\max} \times \Delta T \\ 0.05, & \text{otherwise} \end{cases} \tag{9}$$

Fig. 7. Multiple X-windows for supervised learning.

TABLE II
INITIAL SIMULATION PARAMETERS FOR SUPERVISED LEARNING (UNIT: cm)

$R_v = 20$	$x_{1i} = 40$	$x_{2i} = 40$	$x_{3i} = 120$	$x_{4i} = 120$
$x_{5i} = 120$	$x_{6i} = 200$	$x_{7i} = 200$	$x_{\min} = 30$	$x_{\max} = 220$

provide DVE or IVE visualization. The learning is carried out in a floor plan, as depicted in Fig. 7(a) where R1 represents the mobile robot. Fig. 7(b) depicts the DVE as seen by the robot's top mounted camera. The steering and velocity are controllable through the human-machine interface. When the robot is operated in the VE by a human operator, the five sensor reading and the steering angle of the robot at each sampling step are composed into a data pair (d, y_{2d}) and further compiled into a training set accumulatively. As human's driving behavior is driven by vision, the three-dimensional visualization of the VE may be helpful in transferring human expert's driving skill to the mobile robot.

V. SIMULATION OF RULES LEARNING

A. Supervised Learning Phase

For the supervised learning phase, we assume that the effective range of the ultrasonic sensors is 10 cm–210 cm and the velocity of mobile robot is 15 cm/s. Considering the overhead of the learning algorithm and graphic rendering, a time step of 0.3 s was used. For a maximum angular velocity of 100°/s, the minimum and maximum steering in a time step are $-30°$ and 30°, respectively. This translates to $y_{2\min} = -0.5236$ and $y_{2\max} = 0.5236$. The initial value of b_{2j} is set such that it covers the range $(y_{2\min}, y_{2\max})$ evenly. The initial values of the other parameters used for the simulation are tabulated in Table II. The convergence criteria is $J \leq 1.389 \times 10^{-6}$, which is equivalent to the system's output error of 0.1°. In relation with this, the parameter e depicted in Fig. 5 is chosen to be 0.005, which is equivalent to about 0.3°.

To evaluate the performance of the DR algorithm, the GDR algorithm and the GDRFPA algorithm, simulation runs were implemented under different value of η and α for an input-output pair (d^1, y_{2d}^1), where $d^1 = (43.7, 43.7, 156.2, 156.2, 76.2)$ and $y_{2d}^1 = -0.2094$. Fig. 8 show the learning curves of the three learning algorithms with $\eta = 0.2$ and $\alpha = 0.2$. It can be observed that the GDRFPA has the fastest convergence rate (13 iterations with 75.11 ms) followed by the GDR algorithm (27 iterations with 154.09 ms) and the DR algorithm (53 iterations with 299.95 ms). Simulations also show that the GDRFPA algorithm is not sensitive to initial value of η and α. We adopt the GDRFPA method to train the neural fuzzy system in this paper.

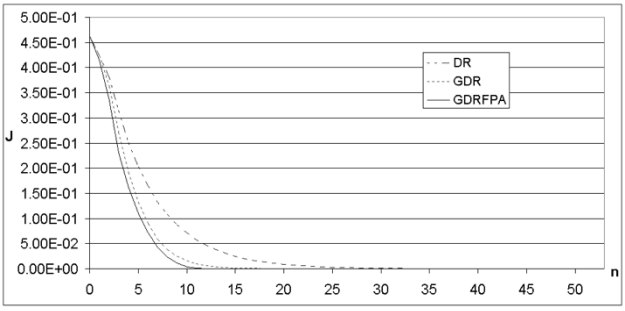

Fig. 8. Learning curves of the three AFLS ($\eta = 0.2, \alpha = 0.2$).

TABLE III
PARAMETERS x_{ri} AFTER ONLINE TRAINING FOR 8000th TRAINING DATA

i \ r	1	2	3	4	5	6	7
1	44.6	50.0	80.5	94.5	100.4	172.5	188.8
2	56.4	58.7	105.8	112.3	117.1	183.6	194.2
3	46.2	50.3	86.9	99.6	106.4	180.5	188.2
4	43.1	46.9	88.4	93.7	100.3	183.2	190.0
5	45.0	49.3	92.2	98.5	102.7	182.9	192.7
	x_{ri}						

The online training scheme [34] was employed for this learning phase. The robot was trained in laboratory L1, L2, and L3. At each configuration, a training data pair (d^m, y_{2d}^m) for $m = 1, \ldots, N$, is composed and the fuzzy system is trained for 20 iterations or until the convergence criterion $J \leq 1.389 \times 10^{-6}$ is met. At the Nth training data, (d^N, y_{2d}^N), the system produces J_m, where $m = 1, \ldots, N$, for each previous data. The average value $J_N = (\sum_{m=1}^{N} J_m)/N$ over the historical data is defined as the ensemble square error (ESE). When the learning terminated at the 8000th training data, the ESE is 0.002 335, i.e., a system output error of 4.1°. The parameters x_{ri} after learning are tabulated in Table III.

B. Reinforcement Learning Phase

1) Rule Learning by the Proposed Method: In this phase, a computer-generated environment (see Fig. 9) is used and the

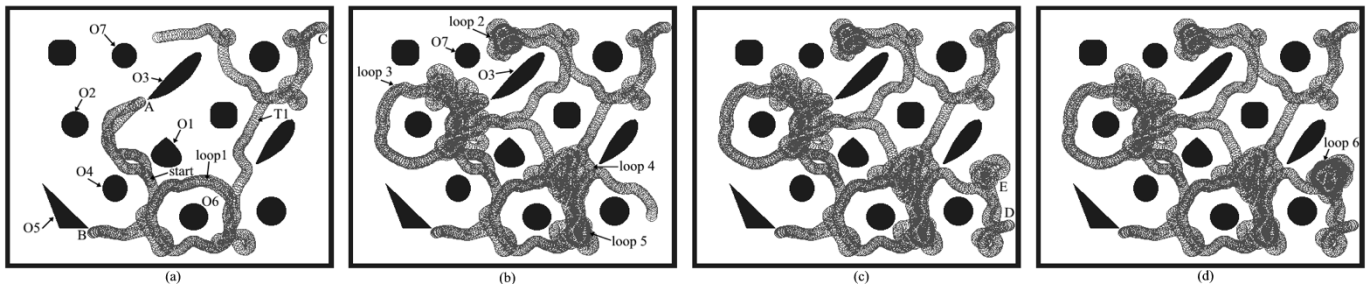

Fig. 9. Simulation of reinforcement learning in a complex environment.

TABLE IV
PARAMETERS USED FOR REINFORCEMENT LEARNING

$a_1 = 3cm/s$	$a_2 = \pi/50$	$\lambda = 0.5$	$b_1 = 15cm/s$	$b_2 = B_{2j}$	$\beta = 0.8$	$\alpha = 0.8$
$\tau = 0.25$	$\delta = 0.85$	$\gamma = 0.95$	$f_1 = 15cm/s$	$f_2 = \pi/5$	$k = 0.2$	$\xi = 1.5$

parameters used for the learning are shown in Table IV, where B_{2j} was learned at the supervised learning phase. In this simulation, a small value of τ is used to ensure sufficient exploration. At the start, $v_{mj}(t)$ are set to some small nonzero values, while $\omega_{mj}(t)$, $\bar{\mu}_j(t)$, $p_m(t-1)$,, and $e_{mj}(t)$ are set to zero. The mobile robot begins at an arbitrary initial configuration with nonzero initial control action. The environment exploration method [17], [18] is employed for training. At each collision, the robot is backtracked four steps and its heading direction is reversed. Details of the training method are referred to in [18]. As the learning proceeds, b_{1j} and b_{2j} are tuned gradually from the initial values to the correct values by (17), which may remove the incorrect rules induced in the supervised learning phase.

As shown in Fig. 9(a), the learning began with a start configuration of (460 cm, 260 cm, 90°)^T. The robot moved forward and encountered the obstacle O_1 on its right. As the rule base had been learned by supervised learning, the robot avoided O_1 successfully and moved toward the obstacle O_2 on its left. It avoided O_2 once again (this agrees with the fact that the prespecified rules can accelerate the learning [17]) and headed on the obstacle O_3 where a collision occurred at A. This demonstrates that the rules are partially correct. The next trial began with the robot reversing its heading and moving toward O_2. After avoiding O_2, it traversed the opening space between O_1 and O_4, where it first avoided O_4 on its right followed by O_1 on its left, until a collision occurred at B. Then a new trial began and the robot went into a loop (loop 1) and was circulating. Instead of repeating the same trajectory as the learning method [17], [18], [34] employing Sutton and Barto's model, the robot took a different path each time circumnavigating around obstacle O_6, because the perturbation generated by the SPG causes a deviation from the previous trajectory. The robot escaped from the loop due to the perturbation and then moved along trajectory T_1 keeping out of the obstacles all the way until a collision at C. Fig. 9(b) demonstrates the new trial after the collision, the robot went into and escaped from loop 2, loop 3, loop 4, and loop 5 one after another and collided with the obstacle at D [see Fig. 9(c)] after which it went into loop 6 [see Fig. 9(d)]. The robot seemed to be trapped by this loop. It might be able to get out of he loop due to the stochastic perturbation; however, to save time, we moved the robot out of it and specified a new configuration manually after it had circulated 20 times. The learning continued until it was terminated at up to 100 000 learning steps.

Due to the stochastic perturbation, a trajectory nearer to the obstacle is more likely to get a collision. Therefore, the robot tends to move with a larger clearance from the obstacles. When the robot entered loop 2, it first circumnavigated in the outer trajectory which is much closer to O_7, then it retreated and moved away from O_7 gradually and finally, orbited in the inner trajectory which roughly kept a same clearance with both O_7 and O_3. This phenomenon is caused by the cooperation of the credit/penalty assignment and perturbation generation; specifically, a move closer to an obstacle is more likely to lead toward a collision hence it is punished and assigned a smaller reinforcement which may move robot away from the obstacle and move the system to a better state, i.e., a larger reinforcement. According to (18), this results in a smaller standard deviation for the stochastic action, i.e., unlikely to get collision. A loop might be induced while the robot is dealing with multiple obstacles as it tends to keep a clearance from all the obstacles. When it goes into a loop, the robot has two possibilities to get rid of it. One is that a large perturbation causes a collision. The other one is that the accumulative deviation from the previous action causes a significant change in the fired rule strengths or causes new rules fired. In this sense, a small value of τ may increase the chance to get out of a loop by the cost of longer learning time. In principle, the robot is always able to get out of the trapped situation provided the learning is not converged and the time is long enough.

2) Comparison With the Related Methods: Firstly, let us consider the existing methods [17], [18], [21] using the original Sutton and Barto's model. We carried out a large number of simulations under the same condition except that the SPG and the supervised learning method was not used. Some typical results are shown in Fig. 10. In Fig. 10(a), the robot began the learning at s_1 with an initial heading of 90°. After a collision with the neighboring boarder, it moved at a straight-line trajectory back and forth with collisions with the obstacles at both ends and could not get out of the trap. The same resulted as it

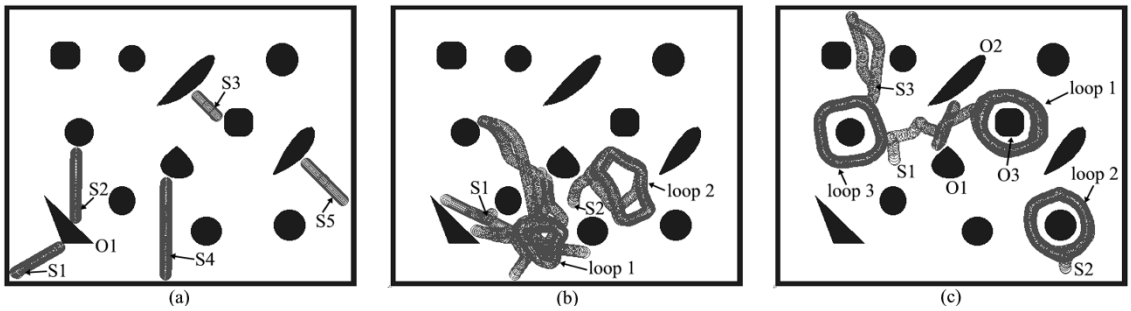

Fig. 10. Simulations without SPG and supervised learning.

was moved to s_2, s_3, s_4, and s_5 (all with 90°initial heading) manually one after another. It is required to restart the learning from scratch. Fig. 10(b) shows a new learning process starting from s_1 with configuration (280 cm, 196 cm, 45°). The robot went into a trap situation (loop1). Then it was moved to a new start s_2 with configuration (520 cm, 240 cm, 90°) manually for a new trial. However, it got into a new trap (loop2) once again.

In Fig. 10(c), the robot started the learning at s_1 with configuration (340 cm, 360 cm, 90°). It collided with obstacle O_1 first and then collided with O_2 and O_1, it then circumnavigated slowly and closely around obstacle O_3 and was trapped by loop1. It went into new traps (loop2 and loop3) again starting from s_2 and s_3. It appears that the learning algorithm thought a convergence has been achieved, although the rule base has not been sufficiently learned. We believe all of these problems are induced by pure exploitation nature of this learning method. Once a reactive behavior to obstacles has been formed, the robot will tend to use the associated rules over and over again hence it repeats the same behavior and refuse to learn further. To overcome this problem, Yung and Ye [18] proposed a new training method, which runs the robot clockwise and counter-clockwise in a narrow corridor-like environment. This overcomes the insufficient learning problem, but the rule learned in such a narrow space is very nearsighted. A better solution is to use the modified model. To prove this point, we carried out a large number of simulation runs using the modified model and have not encountered the above-mentioned problems.

To study the impact of τ, we trained the robot in a corridor-like environment as in [18], except that a bigger corridor width (1 m) was used. In each learning step, the changes of the ASE's weights were calculated by $\Delta\omega_{mj}(t) = \omega_{mj}(t+1) - \omega_{mj}(t)$. The norm of the vector $\Delta\omega_m = (\Delta\omega_{m1}, \Delta\omega_{m2}, \ldots, \Delta\omega_{m243})$, denoted by $||\Delta\omega_m||$, was used to evaluate the strength of exploration as only exploration may result in a significant change in this value. We ran each simulation until 6000th learning step with various value of τ. The results are tabulated in Table V (where SE_1 and SE_2 are the sum of $||\Delta\omega_1||$ and $||\Delta\omega_2||$, respectively, over the 6000 steps), which depicts the average value of SE_1 and SE_2 over 600 simulation runs for each τ. We can observe that a smaller τ may result in stronger exploration strength but more collisions and longer learning time. The plots of the instant reinforcement signal versus learning step show that the leaning with $\tau \geq 0.6$ converged within 6000 learning steps while the others did not. This means an adequate value of τ is required to maintain a tradeoff between exploration and exploitation. We noticed that

TABLE V
EXPLORATION STRENGTH UNDER DIFFERENT τ

	τ=0	τ=0.2	τ=0.4	τ=0.6	τ=0.8	τ=1.0
collision	50	42	18	13	8	5
ave SE_1	4229	3600	3125	2678	1721	1656
ave SE_2	14.59	9.53	7.80	7.11	6.39	5.98

the robot always navigated in the middle of the corridor with a noticeable velocity when the learning converged. This is a significant improvement over the existing methods [17], [18] where the robot may move very slowly in a trajectory very close to either sides of the corridor. We also noticed that the robot moved in a zigzag trajectory in some cases. However, if the mixed learning algorithm was used, no zigzag trajectory was found in all of our simulation runs. This means the mixed learning algorithm may obtain a better rule base.

In summary, due to the curse of dimensionality, reinforcement learning may result in insufficiently learned rule base. The proposed mixed learning algorithm deals with this problem by two steps:

1) it employs the modified Sutton and Barto's model to strengthen the exploration;
2) it starts reinforcement learning from a pretuned rule base, such that the search space is reduced and simplified.

Our simulation results demonstrate that the proposed method outperforms the existing methods.

VI. PERFORMANCE ANALYSIS

The rule base learned in Fig. 9 was tested to be able to perform obstacle avoidance in a number of different environments and was then used by a fuzzy navigator proposed in [18]. The overall performance, such as the motion smoothness, the quality of the navigated path, and the robustness to sensor noise, of the learned rule base are studied using this navigator.

A. Smoothness of Motion

Navigation tasks were carried out in the laboratories other than L1, L2, and L3, as depicted in Fig. 7(a). Fig. 11 depicts a case study of the navigation from s_1 to g_1. For this navigation task, the velocity, acceleration, angular velocity, and angular acceleration of the robot are plotted. It was observed from the plots that 1) the range of acceleration/deceleration is small when the robot passes by an obstacle, but large when the obstacles are in

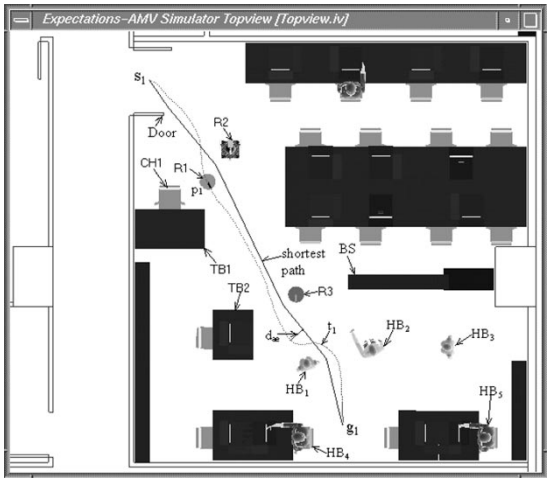

Fig. 11. Navigation from s_1 to g_1 in laboratory L4.

its path; 2) there is no abrupt change of velocity, the acceleration is within (-10 cm/s^2, 15 cm/s^2); and 3) there is no abrupt change in the angular velocity, the angular acceleration is within (-2.0 rad/s^2, 1.5 rad/s^2). These properties have obvious benefits for practical application when the robot's dynamics become an important consideration.

B. Quality of Navigated Path

To evaluate the path achieved by the navigator, the visibility graph method [35] is used to determine the shortest path for each navigation task. For instance, the shortest path determined by this algorithm from s_1 to g_1 is shown by the solid line in Fig. 11. At each time step, the deviation of the robot's position from the shortest path is denoted by d_{ae}. The length of the actual path and the shortest path are represented by p_a and p_e, respectively, and the relative error between the actual path length and the shortest path length $(p_a - p_e)/p_e$ is denoted by E_r. Based on the floor plan, nine navigation tasks were conducted and the results are tabulated in Table VI.

It can be seen that

1) the navigator achieves a path reasonably close to the shortest path;
2) the less obstacles the robot has to tackle, the shortest the path;
3) the relative error and the path deviation are proportional to the number of obstacles.

C. Robustness to Sensor Noise

The robustness to noisy sensor reading is quite important for a navigation algorithm. For a real mobile robot, sensor readings are often noisy, especially in the case that ultrasonic sensors are used. The noise of the sensor may cause incorrect obstacle distances and further cause error in navigation. In the worst case, the noise of the sensor may cause collision. Therefore, a navigation algorithm shall have a large tolerance to noisy sensor data. Due to the limitations of physical experimentation, such as high cost, unrepeatability and damage in the case of collision, simulation is an essential and efficient measure to test the robustness of a navigation algorithm.

TABLE VI
NAVIGATION UNDER VARIOUS OBSTACLE COURSES

T	P_a/cm	p_e/cm	E_r	ave d_{ae}	max d_{ae}	Time	obstacle
1	930.8	878.1	6.0%	15.2cm	40.1cm	55.5s	9
2	884.6	838.5	5.5%	14.0cm	38.0cm	54.8s	8
3	832.3	793.4	4.9%	12.4cm	37.7cm	53.7s	7
4	787.0	754.6	4.3%	10.7cm	33.2cm	53.0s	6
5	706.7	680.8	3.8%	8.6cm	29.3cm	49.9s	5
6	636.2	616.5	3.2%	7.3cm	21.8cm	47.3s	4
7	558.5	545.9	2.3%	5.4cm	12.9cm	43.9s	3
8	495.4	487.1	1.7%	3.9cm	10.3cm	41.5s	2
9	424.1	422.9	1.2%	3.1cm	8.2cm	38.1s	0

TABLE VII
ROBUSTNESS TO SENSOR NOISE

N	SSI (*rad*)	VSI (*cm/s*)	SI
0.0	0.0297	0.29	1.000
0.1	0.0344	0.37	1.000
0.2	0.0390	0.46	1.000
0.3	0.0416	0.57	0.996
0.4	0.0466	0.69	0.990
0.5	0.0523	0.79	0.982
0.6	0.0602	0.98	0.973

Taking into consideration the navigation task from s_1 to g_1, as depicted in Fig. 11, we tested the robustness of the navigator in the presence of various degrees of sensor noise. The simulated sensor noise is assumed to have a Gaussian distribution with mean d; the actual distance and standard deviation of $n \times d$ where $n = 0, 0.1, 0.2, 0.3, 0.3, 0.5$ and 0.6, in this simulation. The robot performed the navigation task 1000 times with each value of n. For measuring the robustness, we use the following definitions.

Definition 1—: Safety Index (SI): $\bar{c}$ The percentage of the simulation runs in which the robot successfully reaches the goal without collision.

Definition 2—: Steering Smoothness Index (SSI): $\bar{\omega} = \sum_{i=1}^{k} |\Delta\bar{\theta}_i|/k$, where $\Delta\bar{\theta}_i$ stands for the absolute average steering angle in the ith simulation run.

Definition 3—: Velocity Smoothness Index (VSI): $\bar{a} = \sum_{i=1}^{k} |\Delta\bar{v}_i|/k$, where $\Delta\bar{v}_i$ stands for the absolute average of the velocity change in the ith simulation run.

The result of the empirical evaluation is summarized in Table VII and plotted in Fig. 12, which shows a degradation of the SI and smoothing index of the proposed navigator as the amount of sensor noise increases. For the smoothness index, the SSI increases only 1.03 times larger while the VSI increases 2.38 times larger, meaning VSI is more sensitive to the sensor noise. As the maximum value of SSI is equivalent to 3.4°, the degradation of SSI is graceful. The maximum value of VSI is 0.98 cm/s, which is also small compared with the maximum velocity of the robot. Even if the value of n is as high as 0.6, the navigator is still able to tackle the obstacle course in most cases. As the standard deviation of sensor measurement for a physical sensor could not be as large as 60% of the actual value in most case, the result means that the navigator has high robustness to noisy sensor data. This is attribute to the proposed modified Sutton and Barto's model. Therefore, the fuzzy system constructed by the proposed learning method has

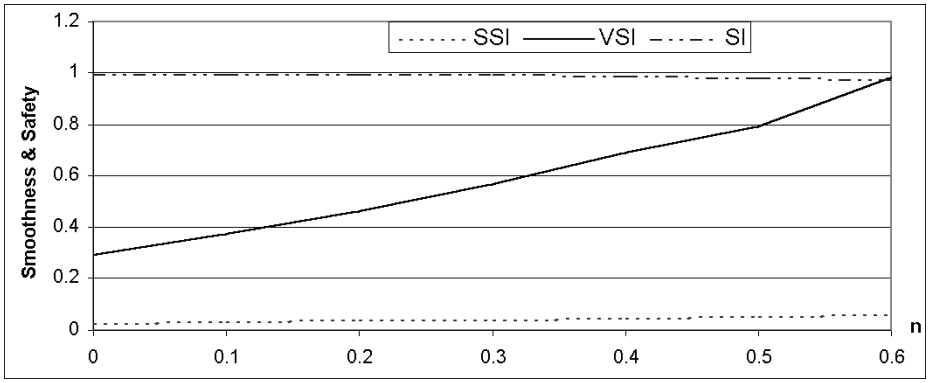

Fig. 12. Smoothness and safety versus noise rate.

high practicality to real robot navigation, in which sensor noise cannot be ignored.

VII. Conclusion

We have presented a neural fuzzy system with mixed learning algorithm where supervised learning method is used to determine the input and output membership functions simultaneously and reinforcement learning algorithm is employed to fine tune the output membership functions. To speed up the supervised learning phase and ensure a stable learning, the GDR method is utilized and fuzzy logic approach is employed to adapt the learning parameters from iteration to iteration. A new learning method using modified Sutton and Barto's model is proposed to ensure a better tradeoff between the exploration and exploitation; therefore, a sufficient and efficient learning is achieved. As the modified model adds a perturbation to the control action before being applied to the robot, the resulted obstacle avoidance module is robust to noisy sensor input. The learning performance under different exploration strength was studied thoroughly and a new method using the norm of the change of the neuron's weights was proposed to evaluate the exploration strength. The simulation results renders that: 1) the GDRFPA learning algorithm is faster than DR method; 2) the modified Sutton and Barto's model has a better exploration hence the robot is able to learn obstacle avoidance more efficiently without human intervention and the learned rule base is robust to perturbation; 3) assisted by the proposed Supervised Learning method, the search space of the Reinforcement Learning is reduced hence the learning is accelerated and it may result in a better rule base; and 4) the mobile robot using the fuzzy system learned by the proposed method is able to perform collision-free navigation. The performance analysis demonstrates that the navigator using the obstacle avoidance module constructed by the proposed learning method features that: 1) it is able to achieve a path reasonably close to the shortest path; 2) it has smooth motion; and 3) it is very robust to sensor noise.

References

[1] B. H. Krogh and D. Feng, "Dynamic generation of subgoals for autonomous mobile robots using local feedback information," *IEEE Trans. Automat. Contr.*, vol. 34, pp. 483–493, May 1989.

[2] N. S. V. Rao, "Robot navigation in unknown generalized polygonal terrains using vision sensors," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 947–962, June 1995.

[3] V. J. Lumelsky *et al.*, "Dynamic path planning in sensor-based terrain acquisition," *IEEE Trans. Robot. Automat.*, vol. 6, pp. 530–540, Aug. 1990.

[4] N. A. V. Rao and S. S. Iyengar, "Autonomous robot navigation in unknown terrains: Incidental learning and environment exploration," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, pp. 1443–1449, Nov./Dec. 1990.

[5] J. Borestein and Y. Koren, "Real-time obstacle avoidance for fast mobile robot," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, pp. 1179–1186, Sept./Oct. 1989.

[6] J. H. Chung and N. Ahuja, "An analytical tractable potential field model of free space and its application in obstacle avoidance," *IEEE Trans. Syst., Man, Cybern. B*, vol. 28, pp. 729–736, Oct. 1998.

[7] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 1398–1404, 1991.

[8] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE J. Robot. Automat.*, vol. RA-2, no. 1, pp. 14–23, 1986.

[9] E. Gat and R. Desai *et al.*, "Behavior control for exploration of planetary surfaces," *IEEE Trans. Robot. Automat.*, vol. 10, pp. 490–503, Aug. 1994.

[10] M. Colombetti and M. Dorigo *et al.*, "Behavior analysis and training-A methodology for behavior engineering," *IEEE Trans. Syst., Man, Cybern. B*, vol. 26, pp. 365–380, June 1996.

[11] J. Donnart and J. Meyer, "Learning reactive and planning rules in a motivationally autonomous animation," *IEEE Trans. Syst., Man, Cybern. B*, vol. 26, pp. 381–395, June 1996.

[12] C. Thorpe, M. Hebert, T. Kanade, and S. Shafer, "Vision and navigation for the Carnegie-Mellon NAVLAB," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 10, pp. 362–373, Mar. 1988.

[13] K. T. Song and J. C. Tai, "Fuzzy navigation of a mobile robot," *Proc. IEEE/RSJ Int. Conf. Intelligent Robotics and Systems*, pp. 621–627, 1992.

[14] E. Tunstel, "Coordination of distributed fuzzy behaviors in mobile robot control," *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, pp. 4009–4014, 1995.

[15] J. Yen and N. Pfluger, "A fuzzy logic based extension to Payton and Rosenblatt's command fusion method for mobile robot navigation," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 971–978, June 1995.

[16] A. Homaifar and E. McCormick, "Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 129–139, May 1995.

[17] H. R. Beom and H. S. Cho, "A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 464–477, Mar. 1995.

[18] N. H. C. Yung and C. Ye, "An intelligent mobile vehicle navigator based on fuzzy logic and reinforcement learning," *IEEE Trans. Syst., Man, Cybern. B*, vol. 29, pp. 314–321, Apr. 1999.

[19] L. Jouffle, "Fuzzy inference system learning by reinforcement methods," *IEEE Trans. Syst., Man, Cybern. C*, vol. 28, pp. 338–355, Sept. 1998.

[20] S. Yamada, M. Nakashima, and S. Shiono, "Reinforcement learning to train a cooperative network with both discrete and continuous output neurons," *IEEE Trans. Neural Networks*, vol. 9, pp. 1502–1508, Nov. 1998.

[21] A. Bonarini, C. Bonacina, and M. Matteucci, "An approach to the design of reinforcement function in real world, agent-based application," *IEEE Trans. Syst., Man, Cybern. B*, vol. 31, pp. 288–301, June 2001.
[22] C. K. Chak, G. Feng, and J. Ma, "An adaptive fuzzy neural network for mimo system model approximation in high-dimensional spaces," *IEEE Trans. Syst., Man, Cybern. B*, vol. 28, pp. 436–446, June 1998.
[23] C. F. Juang and C. T. Lin, "An on-line self-constructing neural fuzzy inference network and its applications," *IEEE Trans. Fuzzy Syst.*, vol. 6, pp. 12–32, Feb. 1998.
[24] C. T. Lin and C. S. G. Lee, "Neural-network-based fuzzy logic control and decision system," *IEEE Trans. Comput.*, vol. 40, no. 12, pp. 1320–1336, 1991.
[25] ——, "Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems," *IEEE Trans. Fuzzy Syst.*, vol. 2, pp. 46–63, Feb. 1994.
[26] C. J. Lin and C. T. Lin, "Reinforcement learning for an art-based fuzzy adaptive learning control network," *IEEE Trans. Neural Networks*, vol. 7, pp. 709–731, May 1996.
[27] C. Ye and N. H. C. Yung, "Vehicle navigation strategy based on behavior fusion," *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, pp. 3698–3703, 1997.
[28] S. Haykin, *Neural Networks: A Comprehensive Foundation*. New York: Macmillan, 1994.
[29] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Netw.*, vol. 1, no. 4, pp. 295–307, 1988.
[30] P. Arabshahi, J. J. Choi, R. J. Marks, II, and T. P. Caudell, "Fuzzy parameter adaptation in optimization: Some neural net training examples," *IEEE Comput. Sci. Eng.*, vol. 3, no. 1, pp. 57–65, Spring 1996.
[31] *Handbook of Intelligent Control*, Van Nostrand, New York, 1992, pp. 527–559. The role of exploration in learning control.
[32] H. R. Berenji and P. Khedkar, "Learning and tuning fuzzy logic controllers through reinforcements," *IEEE Trans. Neural Networks*, vol. 3, pp. 724–740, Sept. 1992.
[33] N. H. C. Yung and C. Ye, "Expectations-An autonomous mobile vehicle simulator," *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, pp. 2290–2295, 1997.
[34] C. Ye, "Behavior-based fuzzy navigator of mobile vehicle in unknown and dynamically changing environment," Ph.D. dissertation, Dept. EEE, University of Hong Kong, Hong Kong, 1999.
[35] J. C. Latombe, *Robot Motion Planning*. Norwell, MA: Kluwer, 1991.

Cang Ye (S'97–M'00) received the B.E. and M.E. degrees from the University of Science and Technology of China, Hefei, Anhui, in 1988 and 1991, respectively, and the Ph.D. degree from the University of Hong Kong, Hong Kong, in 1999.

He was a Senior Research Engineer at the Laboratory for Intelligent Transportation System Research, University of Hong Kong, from 1998 to 1999, and a Research Fellow at the School of Electronic and Electrical Engineering, Nanyang Technological University, Nanyang, Singapore, from 1999 to 2001. He is currently with the Advanced Technologies Laboratory, The University of Michigan, Ann Arbor. He serves as a reviewer for numerous reputable international journals.

Dr. Ye is listed in *Marquis Who's Who in the World.*

Nelson H. C. Yung (S'82–M'85–SM'96) received the B.Sc. and Ph.D. degrees from the University of Newcastle upon Tyne, Newcastle upon Tyne, U.K.

He was Lecturer at the same university from 1985 to 1990. From 1990 to 1993, he was a Senior Research Scientist at the Department of Defense in Australia. In 1993, he joined the University of Hong Kong as Associate Professor. He currently leads a research team in digital image processing and intelligent transportation systems. He is the Founding Director of the *Laboratory for Intelligent Transportation Systems Research* at Hong Kong University. He has coauthored a computer vision book, and has published over 100 journal and conference papers in the areas of digital image processing, parallel algorithms, visual traffic surveillance, and autonomous vehicle navigation. He serves as a reviewer for numerous international journals.

Dr. Yung is a Member of the Advisory Panel of the *ITS Strategy Review*, Transport Department, HKSAR. He is a Chartered Electrical Engineer and Member of the HKIE and IEE. He was a Croucher Scholar and his team won the Silver Award from the HKEIA for Outstanding Innovation and Technology Product, 2000, for the MOVER solution. He is listed in *Marquis Who's Who in the World.*

Danwei Wang (S'88–M'89) received the B.E. degree from the South China University of Technology in 1982 and the M.S.E. and Ph.D. degrees from The University of Michigan, Ann Arbor, in 1985 and 1989, respectively.

Since 1989, he has been with the Nanyang Technological University, Nanyang, Singapore, where he is an Associate Professor and Deputy Director of the Robotics Research Center. He has served as General Chairman and Technical Chairman in international conferences (ICARCV and ACCV'95). He has published more than 100 technical articles in the areas of iterative learning control, robust control, and adaptive control systems, as well as manipulator/mobile robot dynamics, path planning, and control.

Dr. Wang is a Member of the IEEE Singapore Robotics and Automation Chapter. He was a recipient of an Alexander von Humboldt Fellowship, Germany. He is Associate Editor of Conference Editorial Board, IEEE Control Systems Society.