Neurofuzzy-Based Approach to Mobile Robot Navigation in Unknown Environments

Anmin Zhu and Simon X. Yang, *Member, IEEE*

***Abstract*—In this paper, a neurofuzzy-based approach is proposed, which coordinates the sensor information and robot motion together. A fuzzy logic system is designed with two basic behaviors, target seeking and obstacle avoidance. A learning algorithm based on neural network techniques is developed to tune the parameters of membership functions, which smooths the trajectory generated by the fuzzy logic system. Another learning algorithm is developed to suppress redundant rules in the designed rule base. A state memory strategy is proposed for resolving the "dead cycle" problem. Under the control of the proposed model, a mobile robot can adequately sense the environment around, autonomously avoid static and moving obstacles, and generate reasonable trajectories toward the target in various situations without suffering from the "dead cycle" problems. The effectiveness and efficiency of the proposed approach are demonstrated by simulation studies.**

***Index Terms*—"Dead cycle" problem, mobile robot navigation, neurofuzzy, parameter tuning, redundant rule suppression.**

I. Introduction

MOBILE robot navigation is an essential issue in robotics and artificial intelligence. Real-time navigation is an easy task for human beings or animals, but very difficult for robots, especially in unknown and changing environments. For real-time autonomous navigation, the robot should be capable of sensing its environment, interpreting the sensed information to obtain the knowledge of its location and the environment, planning a real-time trajectory from an initial position to a target with obstacle avoidance, and controlling the robot direction and velocity to reach the target.

Neural network-based approaches have been employed for robot motion planning. In these approaches, the robot is treated as a point moving under the influence of an artificial neural potential field. The attractive potential force attracts the robot toward the target configuration, while repulsive potential forces push it away from obstacles. Dahm *et al*. [2], [4] used a neural field approach described by an integrodifferential equation, which can be discretized to obtain a nonlinear competitive dynamical system affecting a set of artificial neurons. The next movement step of the robot depends on a neural dynamics mechanism, which actively selects a movement direction from a set of possible directions. But the robot path is not continuous and the kinematic constraint of the robot is not taken into account in this algorithm. Yang and Meng [25]–[27] proposed a neural network architecture, which is a discrete topologically organized map, for robot motion planning using a shunting neural network model. They discretize the environment and use neural activity to represent the environment information about the target and obstacles. However, the generated trajectory is a discrete one and the robot speed is considered as a constant. In addition, most of these models assume that the whole workspace is definitely known by the robot, that is impracticable in real robot control.

Fuzzy logic control is well suited for controlling a mobile robot because it is capable of making inferences even under uncertainty [17]. For mobile robot navigation, fuzzy logic approaches have been investigated by several researchers. Li and Yang [9] proposed an obstacle avoidance approach using fuzzy logic, but the input sensors are separately inferred, and only a few simple cases are shown in the paper. Lee and Wang [8] proposed a collision-avoidance approach using fuzzy logic, where different modules, such as avoiding-static-obstacle module, avoiding-moving-obstacle module, and directing-toward-target module, are created for the robot navigation. However, these modules are separately inferred and are not as coordinated as human reasoning. Xu and Tso [24] proposed a reactive behavior-based fuzzy logical controller with a virtual target technique called "local target switching," which utilizes the rules to define the robot reaction to unknown environments and applies fuzzy inference to coordinate different reactive behaviors. However, the "dead cycle" problem (i.e., going around in circles or cycling between multiple traps) may still occur in some cases (e.g., just assume that the target is located at the top right corner instead of above the workspace in Fig. 7 in [24]. Saffiotti *et al*. proposed some fuzzy logic methods for robot navigation (e.g., [16] and [18]). However, these methods cannot guarantee that the robot will not be trapped on local minima or infinite loops. The process of tuning the parameters of fuzzy rules may be rather difficult. Yang *et al*. [28] developed a navigation algorithm for a mobile robot system by combining a fuzzy logic architecture with a virtual centrifugal effect algorithm (VCEA). In this model, the goal seeking subproblem and obstacle avoidance subproblem are solved by two separate fuzzy logic systems. The VCEA is developed to go around an obstacle by temporarily forgetting the target. However, this algorithm focuses on direction control without considering velocity control. Furthermore, it cannot solve the "dead cycle" problem in an U-shaped obstacle environment. Vadakkepat *et al*. [22] proposed a fuzzy behavior-based model to control a team of three soccer robots to finish a simple task in a court environment. Four rules, 12 behaviors,

Manuscript received August 19, 2005; revised February 18, 2006. This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada. This paper was recommended by Associate Editor F.-Y. Wag.

A. Zhu is with the College of Information Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: azhu@szu.edu.cn).

S. X. Yang is with the Advanced Robotics and Intelligent Systems (ARIS) Laboratory, School of Engineering, University of Guelph, Guelph, ON N1G 2W1, Canada (e-mail: syang@uoguelph.ca).

Digital Object Identifier 10.1109/TSMCC.2007.897499

and a set of actions were designed. Fuzzy logic was used to implement individual behaviors, to coordinate the various behaviors, to select roles for each robot, and for robot perception, decision-making, and speed control. Instead of modeling the environment using a high level of accuracy, Aguirre and Gonzalez [1] proposed a perceptual model based on fuzzy logic in a hybrid deliberative-reactive architecture. This model improved the performance in the two aspects of robot navigation: perception and reasoning. Fuzzy logic is used in different parts of the perceptual model. However, the model focuses on map building, and thus is computationally expensive.

To improve the performance, some neurofuzzy methods are used. Godjevac [5] proposed a neurofuzzy model for a mobile robot to avoid obstacles. More than 600 rules are formulated, where many of them are redundant and there are no methods to suppress the useless rules. Godjevac and Steele [6] later developed a neurofuzzy controller with a learning procedure, which is applied to the obstacle avoidance and wall following for a KHEPERA mobile robot. With only a few simple cases shown in the paper, it seems that the "dead cycle" problems cannot be resolved. Marichal *et al.* [10] presented another neurofuzzy controller by a three-layer neural network with a competitive learning algorithm for a mobile robot. It automatically extracts the fuzzy rules and the membership functions through a set of trajectories obtained from human guidance. Because it is difficult to determine the fuzzy rules for complex environments with obstacles, this model is suitable for very simple environments. Song and Sheen [20] developed a heuristic fuzzy-neural network using a pattern-recognition approach. This approach can reduce the number of rules by constructing the environment (e.g., obstacles) using several prototype patterns. It is suitable for simple environments, because the more complex the environment, the more difficult to construct the patterns. Hagras *et al.* [7] developed a converging online-learning genetic algorithm mechanism for learning the membership functions of individual behaviors of an autonomous mobile robot. In that approach, a hierarchical fuzzy controller is used to reduce the number of rules, while the genetic algorithm is applied to tune the parameters of the membership functions. The robot needs to be equipped with a short-time memory to store the previous 6000 actions and the robot has to go back to some previous positions to evaluate a new solution. Rusu *et al.* [14] proposed a neurofuzzy controller for mobile robot navigation in indoor environments. Infrared and contact sensors are used for detecting target and avoiding collisions. Two levels with several behaviors are designed for the controller. Fuzzy inference is used in every behavior. A neural network is used to tune the system parameters. A switching coordination technique is used to select the appropriate behavior. Command fusion is used to combine the output of several neurofuzzy subsystems. However, the design of the rule base for the controller is not clear. The meanings of system parameters are vague when being trained by a neural network. Furthermore, there is no evidence that the controller is able to resolve the "dead cycle" problem.

To resolve the "dead cycle" problem, some methods have been proposed, such as "bug algorithm" and its extensions [11], [12], [19], [29]. The main idea of "bug algorithm" [19] is to draw a virtual line from the start position of the robot to the target and define the hit points and leave points when the robot is meeting or leaving the virtual line, and then to decide if the robot should move toward the target or move along the obstacle, by memorizing the hit points and leaving points, and by comparing the distances between the robot and the hit points to the target. Some extended algorithms [29] do not consider the distances, but add some counters, angle, or topology of the obstacles. A theoretical proof of the dead cycle free was given in [29]. The worst path length and average path length are theoretically evaluated in [11] and [12], but in these algorithms, the robot is considered as a point, and the robot velocity is not considered at all. Furthermore, these algorithms have to memorize the hit and leave points.

In this paper, a novel neurofuzzy-based model is presented for reactive navigation of mobile robots in unknown environments. The inputs of the fuzzy controller are the outputs from the multisensor system, including the obstacle distances obtained from the left, front, and right sensor groups, the target direction, and the current robot speed. A set of linguistic fuzzy rules are developed to implement expert knowledge under various situations. The output signals from the fuzzy controller are the accelerations of left and right wheels, respectively. A learning algorithm based on neural network techniques will be developed to tune the parameters of the membership functions. Another learning algorithm is developed to autonomously suppress the redundant fuzzy rules. A state memory strategy is proposed for resolving the "dead cycle" problem. Under the control of the proposed neurofuzzy-based model, the mobile robot can generate reasonable trajectories toward the target in various situations without suffering from the "dead cycle" problems.

This paper is organized as follows: the proposed approach to reactive navigation is presented in Section II, including the architecture of the navigation system, the design of the fuzzy controller, the learning algorithm to tune the model parameters, the learning algorithm to suppress redundant rules, and the state memorizing strategy to resolve the "dead cycle" problem; Section III provides simulation studies; and finally some concluding remarks are given in Section IV.

II. Proposed Approach

In this section, the overall control structure of the proposed neurofuzzy model is first described. The fuzzy controller is then designed. After that, the neural network-based learning algorithms are developed to tune the model parameters, and to suppress redundant rules. Finally, the state memorizing strategy to resolve the "dead cycle" problem is presented.

A. Overall Control Structure

To control a mobile robot to reach its destination with obstacle avoidance, sensors must be mounted on the robot to sense the environment and interpret the sensed information. The main sensors of the mobile robot are shown in Fig. 1. The robot is employed to test the proposed fuzzy logic-based system. The robot has two front coaxle wheels driven by different motors separately, and a third passive omnidirectional caster. Through

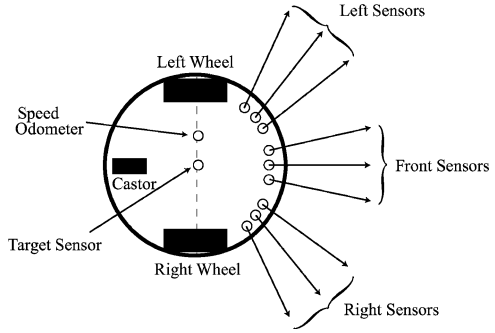

Fig. 1. Model of a mobile robot with sensors.

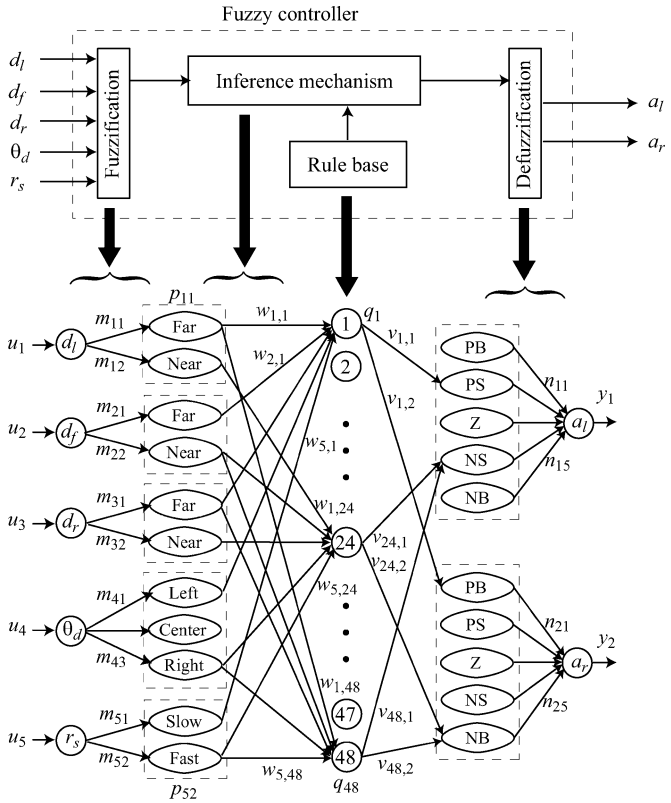

Fig. 2. Block diagram of the proposed fuzzy controller. m_{ij}, the centers of MFs for input variables; n_{ls}, the centers of MFs for the output variables; p_{ij}, the degree of Memberships; q_k, conjunction degree of IF part of rules; $w_{i,k}$, weights related to m_{ij}; $v_{k,l}$, weights related to n_{ls}; PB, positive big; PS, positive small; Z, zero; NS, negative small; NB, negative big.

adjusting the accelerations of the two driven wheels, respectively, the velocity and motion direction of the mobile robot can be controlled. For the reactive navigation to be easily realized, nine ultrasonic sensors are incorporated on the robot, so that the distances between the robot and the obstacles can be measured. These sensors are equipped on the left, the right, and in the middle of the front part of the robot to cover a semicircular area around the front half of the robot and to protect the robot from collisions. The nine sensors are divided into three groups (each group has three sensors) to measure the distance from obstacles to the left, right, and front of the robot. In order to reach a target, a simple optical range finder with a directing beam and a rotating mirror [3], or a global positioning system (GPS), would be used to obtain the target direction. A speed odometer is equipped on the robot to measure the current robot speed. The mission of the robot is to navigate in the unknown and dynamic environment from an initial position to a desired target with obstacle avoidance, without trapping in any trap area.

The robot motion is controlled by adjusting the accelerations of two driven wheels. At first, the robot system has to judge if the distance is "far" or "close," if the speed is "fast" or "slow," and so on, and then decide the motion commands. These informations ("far," "close," "fast," "slow," etc.) are uncertain and imprecise. They are difficult to be represented by conventional logic systems, or mathematical methods for robot systems. But they are easy for human beings to use. Usually people do not need precise, numerical information input to make a decision, but they are able to perform highly adaptive control, because there is a "fuzzy" concept in human knowledge. Fuzzy logic is known to be an organized method for dealing with imprecise knowledge. Using linguistic rules, the fuzzy logic system mimics human decision making to deal unclearly expressed concepts, to deal with imprecise or imperfect information, and to improve knowledge representation and uncertainty reasoning. Therefore, a fuzzy logic method is selected to deal with the sensor-based robot motion control problem.

Fuzzy logic offers a framework for representing imprecise, uncertain knowledge. They make use of human knowledge in the form of linguistic rules. But the disadvantages are that fuzzy logic needs highly abstract heuristics, needs experts for rule discovery with data relationships, especially, as it lacks of self-organizing and self-tuning mechanisms. This results in difficulties to decide the parameters of membership functions. Another drawback is the lack of a systematic procedure to transform expert knowledge into the rule base. This results in many redundant rules in the rule base. On the other hand, neural networks are nonmodel systems that are organized in a way to simulate the cells of human brain. They learn from the underlying relationships of data. Neural networks have self-learning capability, self-tuning capability, without the need to know data relationship, and can be used to model various systems. Therefore, fuzzy logic and neural networks can be combined to solve the complex robot navigation control problem and improve the performance.

The structure of the proposed neurofuzzy approach is shown in Fig. 2. This is a five-layer neurofuzzy network. The inputs of the fuzzy controller are the outputs from the multisensor system: the obstacle distances d_l, d_f, d_r obtained from the left, front, and right sensor groups, the target direction θ_d (that is, the angle between the robot moving direction and the line connecting the robot center with the target), and the current robot speed r_s. The second layer denotes the terms of input membership variables. The third layer denotes the rule base. The fourth layer denotes the terms of output membership variables. In the fifth layer, the output signals from the fuzzy controller are the accelerations of left and right wheels, a_l and a_r, respectively.

There are differences between the proposed approach and most conventional neurofuzzy methods (e.g., [5], [7], [10], [14] and [20]).

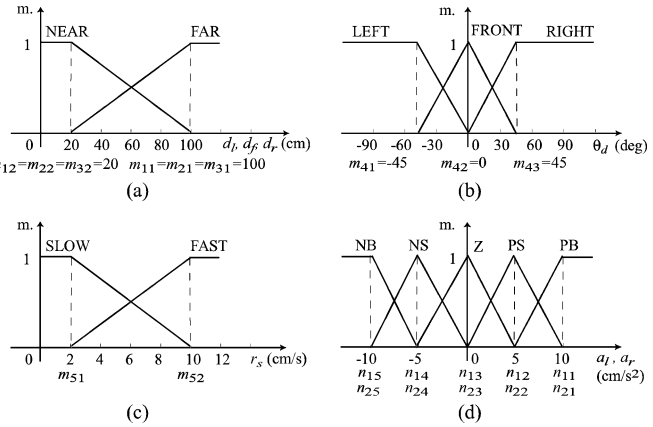

Fig. 3. Membership functions of the input and output variables. m_{ij}, n_{ls}, the centers of membership functions; m., membership; meaning of NB to PB, see the foot of Fig. 4. (a) of obstacle distances, (b) of the target direction, (c) of the current robot speed, (d) of accelerations.

IF

obstacle distance on left (d_l) is NEAR and

obstacle distance on front (d_f) is FAR and

obstacle distance on right (d_r) is FAR and

target direction (θ_d) is RIGHT and

current robot speed (r_s) is SLOW

THEN

acceleration of left wheel (a_l) is PB

acceleration of right wheel (a_r) is PS

Fig. 4. Example of the inference rules. $d_l = \text{NEAR}, \{\text{FAR}\}$, $d_f = \{\text{NEAR}, \text{FAR}\}$, $d_r = \{\text{NEAR}, \text{FAR}\}$, $\theta_d = \{\text{LEFT}, \text{CENTER}, \text{RIGHT}\}$, $r_s = \{\text{SLOW}, \text{FAST}\}$, $a_l = \{\text{PB}, \text{PS}, \text{Z}, \text{NS}, \text{NB}\}$, $a_r = \{\text{PB}, \text{PS}, \text{Z}, \text{NS}, \text{NB}\}$.

1) Much less rules are needed, while some conventional methods need a large number of rules (e.g., 600 in [5]). This simplifies the structure of the neurofuzzy model by reducing the number of the hidden layer neuron, and reduces the computational time.
2) The physical meaning of the parameters remains the same during the training, which is lost in conventional neurofuzzy methods [10].
3) The neural network-based methods are developed to improve the performance in the proposed approach.
4) A state memorizing strategy is designed to resolve the "dead cycle" problem in the proposed approach.

B. Design of the Fuzzy Logic Controller

The proposed fuzzy logic method is simple, easy to understand, has human-like intelligence, and quick reaction capability. Fuzzification, inference mechanism, and defuzzification are considered to create the fuzzy logic controller.

1) Fuzzification: The fuzzification procedure maps the crisp input values to the linguistic fuzzy terms with membership values between 0 and 1. In most fuzzy logic systems, nonfuzzy input data are mapped to fuzzy sets by treating them as Gaussian, triangle, trapezoid, sharp peak membership functions, etc. In this paper, triangle functions, S-type and Z-type functions will be chosen to represent fuzzy membership functions. Membership functions for the terms of the input and output variables in this controller are shown in Fig. 3.

The outputs of the fuzzification procedure are given as follows, for a triangle function:

$$p_{ij} = \begin{cases} 1 - \frac{2|u_i - m_{ij}|}{\sigma_{ij}}, & \text{if } m_{ij} - \frac{\sigma_{ij}}{2} < u_i < m_{ij} + \frac{\sigma_{ij}}{2} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

for an S-type function:

$$p_{ij} = \begin{cases} 0, & \text{if } u_i < m_{ij} - \frac{\sigma_{ij}}{2} \\ 1, & \text{if } u_i > m_{ij} \\ 1 - \frac{2|u_i - m_{ij}|}{\sigma_{ij}}, & \text{otherwise} \end{cases} \tag{2}$$

for a Z-type function:

$$p_{ij} = \begin{cases} 0, & \text{if } u_i > m_{ij} + \frac{\sigma_{ij}}{2} \\ 1, & \text{if } u_i < m_{ij} \\ 1 - \frac{2|u_i - m_{ij}|}{\sigma_{ij}}, & \text{otherwise} \end{cases} \tag{3}$$

where $i = 1, 2, \ldots, 5$, represents the number of input signals; $j = 1, 2, \ldots, 5$, is the number of terms of the input variables; p_{ij} is the degree of membership for the ith input corresponding to the jth term of the input variable; u_i is the ith input signal to the fuzzy controller, $\{u_1, u_2, u_3, u_4, u_5\} = \{d_l, d_f, d_r, \theta_d, r_s\}$; m_{ij} is the center of the membership function corresponding to the ith input and the jth term of the input variable; and σ_{ij} is the width of the membership function corresponding to the ith input and the jth term of the input variable. For example, $u_4 = t_d$, represents the input value about the target direction; $m_{42} = 0$ means the value of the membership function center related to the second term ("Center") of the fourth input variable (u_4 or t_d) is 0; and $\sigma_{42} = 90$ is the width of the membership function.

2) Inference Mechanism: The inference mechanism is responsible for decision making in the fuzzy controller using approximate reasoning. The rule base is essential for the controller, which stores the rules governing the input and output relationship of the proposed controller. The inference rules are in the general form in Fig. 4.

Forty-eight rules are formulated for the proposed controller given in Table I. There are two basic behaviors in the rule base: target seeking behavior, and obstacle avoidance behavior. In general, the target seeking behavior is used to change the direction of the robot toward the target when there are no obstacles blocking the robot. The obstacle avoidance behavior is used to turn away from the obstacles disregarding the direction of the target when obstacles are close to the robot.

3) Defuzzification: The defuzzification procedure maps the fuzzy output from the inference mechanism to a crisp signal. There are many methods that can be used to convert the conclusion of the inference mechanism into the actual output of the fuzzy controller. The "center of gravity (CoG)" method is used in the proposed controller, which combines the outputs represented by the implied fuzzy sets from all rules to generate

TABLE I
RULE BASE OF THE PROPOSED FUZZY LOGIC CONTROLLER

rule No.	input					output		behav.	redun. rule
	d_l	d_f	d_r	θ_d	r_s	a_l	a_r		
1	F	F	F	L	SL	PS	PB	TS	
2	F	F	F	L	FS	NS	Z	TS	
3	F	F	F	C	SL	PB	PB	TS	
4	F	F	F	C	FS	Z	Z	TS	
5	F	F	F	R	SL	PB	PS	TS	
6	F	F	F	R	FS	Z	NS	TS	
7	F	F	N	L	SL	Z	PS	TS	
8	F	F	N	L	FS	NS	Z	TS	
9	F	F	N	C	SL	Z	PS	OA	
10	F	F	N	C	FS	NS	Z	OA	
11	F	F	N	R	SL	NS	Z	OA	
12	F	F	N	R	FS	NB	NS	OA	*
13	F	N	N	L	SL	NS	Z	OA	
14	F	N	N	L	FS	NB	NS	OA	*
15	F	N	N	C	SL	NS	Z	OA	
16	F	N	N	C	FS	NB	NS	OA	*
17	F	N	N	R	SL	NS	Z	OA	
18	F	N	N	R	FS	NB	NS	OA	*
19	N	N	N	L	SL	NS	Z	OA	
20	N	N	N	L	FS	NB	NS	OA	*
21	N	N	N	C	SL	NS	Z	OA	
22	N	N	N	C	FS	NB	NS	OA	*
23	N	N	N	R	SL	Z	NS	OA	
24	N	N	N	R	FS	NS	NB	OA	
25	N	F	N	L	SL	Z	Z	OA	
26	N	F	N	L	FS	NS	NS	OA	
27	N	F	N	C	SL	PS	PS	TS	
28	N	F	N	C	FS	NS	NS	TS	
29	N	F	N	R	SL	Z	Z	OA	
30	N	F	N	R	FS	NS	NS	OA	
31	N	F	F	L	SL	Z	NS	OA	
32	N	F	F	L	FS	NS	NB	OA	
33	N	F	F	C	SL	PS	Z	OA	
34	N	F	F	C	FS	Z	NS	OA	*
35	N	F	F	R	SL	PS	Z	TS	
36	N	F	F	R	FS	Z	NS	TS	
37	N	N	F	L	SL	Z	NS	OA	
38	N	N	F	L	FS	NS	NB	OA	
39	N	N	F	C	SL	Z	NB	OA	
40	N	N	F	C	FS	NS	NB	OA	*
41	N	N	F	R	SL	Z	NS	OA	
42	N	N	F	R	FS	NS	NB	OA	
43	F	N	F	L	SL	NS	Z	OA	
44	F	N	F	L	FS	NB	NS	OA	*
45	F	N	F	C	SL	NS	Z	OA	
46	F	N	F	C	FS	NB	NS	OA	*
47	F	N	F	R	SL	Z	NS	OA	
48	F	N	F	R	FS	NS	NB	OA	

N, near; F, far; L, left; C, center; R, right; SL, slow; FS, fast; TS, target seeking; OA, obstacle avoidance.

the gravity centroid of the possibility distribution for a control action. The value of the output variables a_l and a_r are given as

$$a_l = \frac{\sum_{k=1}^{48} v_{k,1} q_k}{\sum_{k=1}^{48} q_k} \tag{4}$$

$$a_r = \frac{\sum_{k=1}^{48} v_{k,2} q_k}{\sum_{k=1}^{48} q_k} \tag{5}$$

$$q_k = \min\{p_{1k_1}, p_{2k_2}, p_{3k_3}, p_{4k_4}, p_{5k_5}\} \tag{6}$$

where $v_{k,1}$ and $v_{k,2}$ denote the estimated values of the outputs provided by the kth rule, which are related to the center of membership functions of the output variables, q_k is the conjunction degree of the IF part of the kth rule, k is the number of rules, p_{ik_1} is the degree of the membership for the ith input contributing to the kth rule, and i is the number of input values. Different from other methods [13], [15], [21], [30], in which only a behavior is active or unvaried weights are set to the behaviors, in the proposed model, all of the behaviors are combined through the fuzzy logic defuzzification procedure.

C. Physical Meanings of Variables and Parameters

The proposed approach keeps the physical meanings of the variables and parameters during the processing, while the conventional fuzzy logic-based model [10] cannot keep the physical meanings of variables.

The physical meaning of fuzzy variables is explained in Fig. 2. In this paper, index $i = 1, \cdots, 5$ represents the number of inputs, $j = 1, \cdots, 5$ represents the number of the terms of the input variables, $k = 1, \cdots, 48$ represents the number of rules, $l = 1, 2$ represents the number of the outputs, $s = 1, \cdots, 5$ represents the number of terms of the output variables, and $u_1, u_2, u_3, u_4, u_5 = d_l, d_f, d_r, \theta_d, r_s$ is the input vector, $y_1, y_2 = a_l, a_r$ is the output vector, described in (1) and (5). The variable p_{ij} is the degree of membership for the ith input corresponding to the jth term of the input variable obtained by (1) according to different membership functions. The variable q_k is conjunction degree of the IF part of the kth rule obtained by (6). The variable $w_{i,k}$ denotes the center of the membership function corresponding to the ith input and the kth rule, which can be assigned to one of the m_{ij} according to the rule base, e.g., $w_{1,1} = m_{11}, w_{2,1} = m_{21}, w_{5,1} = m_{51}, w_{1,24} = m_{,2}$, $w_{5,1} = m_{52}, w_{1,48} = m_{11}$, and $w_{5,48} = m_{52}$. The variables $v_{k,l}$ are the estimated value of the outputs provided by the kth rule, which are related to one of the center of membership functions of the output variables. Assume n_{ls} denote the centers of the membership functions of variables a_l and a_r. Assume the width of the membership functions are constant (e.g., 1). Then $v_{1,1} = n_{12}, v_{1,2} = n_{21}, v_{24,1} = n_{14}, v_{24,2} = n_{25}, v_{48,1} = n_{14}$, and $v_{48,2} = n_{25}$.

D. Algorithm to Tune the Model Parameters

To smooth the trajectory generated by the fuzzy logic model, a learning algorithm based on a neural network technique is developed. The effect of the output variables mainly depends on the center of the membership functions when the rule base is designed. The widths of the membership functions can be disregarded, and can usually be set to constant values. The membership function center values of the input and output variables may be improved by the neural network learning property. The vector of 21 parameters to be tuned in the proposed model is set as

$$Z = \{m_{11}, m_{12}, m_{21}, m_{22}, m_{31}, m_{32}, m_{41}, m_{42}, m_{43}, m_{51}, m_{52}, n_{11}, n_{12}, n_{13}, n_{14}, n_{15}, n_{21}, n_{22}, n_{23}, n_{24}, n_{25}\}. \tag{7}$$

One of the most widely used algorithms is the least-mean square (LMS) algorithm based on the idea of stochastic approximation method to adjust parameters of an application system. There are errors between the desired output and the actual output of the system. Using the errors, the LMS algorithm adjusts the system parameters, thus, altering its response characteristics by minimizing a measure of the error, thereby closing the

performance loop. In this paper, the LMS algorithm is used to minimize the following criterion function:

$$E = \frac{1}{2}\sum_{l=1}^{2}(y_l - \hat{y}_l)^2 \tag{8}$$

where $\{y_1, y_2\} = \{a_l, a_r\}$ is the output vector; $\{\hat{y}_1, \hat{y}_2\}$ is the desired output vector obtained by derivation of the desired trajectory that is manually marked on the simulator. Thus, the parameters would be adapted as

$$Z(t+1) = Z(t) - \varepsilon\frac{\partial E}{\partial Z} \tag{9}$$

where Z is the parameter vector to adapt; ε is the learning rate; $Z(t)$ is the parameter vector Z at time t; and t is the number of iterations. Thus, the equations for the adaptation of the parameters are given as

$$m_{ij}(t+1) = m_{ij}(t) - \varepsilon_m\frac{\partial E}{\partial m_{ij}}, \quad i = 1, \ldots, 5, j = 1, 2, 3 \tag{10}$$

$$n_{ls}(t+1) = n_{ls}(t) - \varepsilon_n\frac{\partial E}{\partial n_{ls}}, \quad l = 1, 2, s = 1, \ldots, 5 \tag{11}$$

where ε_m and ε_n are the learning rates. Therefore, it is only necessary to calculate the partial derivative of the criterion function with respect to the particular parameter to get the expression for each of them. The corresponding expressions for the membership function centers of input and output variables are given as

$$\begin{aligned}\frac{\partial E}{\partial m_{ij}} &= \sum_{l=1}^{2}\left(\frac{\partial E}{\partial y_l}\frac{\partial y_l}{\partial q_k}\frac{\partial q_k}{\partial p_{ij}}\frac{\partial p_{ij}}{\partial m_{ij}}\right) \\ &= -2\sum_{l=1}^{2}\left[(y_l - \hat{y}_l)\frac{v_{k,l}\sum_{k=1}^{48} q_k - \sum_{k=1}^{48} v_{k,l}q_k}{(\sum_{k=1}^{48} q_k)^2}\right] \\ &\quad \times \frac{\operatorname{sign}(u_i - m_{ij})}{\sigma_{ij}}\end{aligned} \tag{12}$$

$$\frac{\partial E}{\partial n_{ls}} = \sum_{l=1}^{2}\left(\frac{\partial E}{\partial y_l}\frac{\partial y_l}{\partial v_{k,l}}\frac{\partial v_{k,l}}{\partial n_{ls}}\right) = (y_l - \hat{y}_l)\frac{q_k}{\sum_{k=1}^{48} q_k}. \tag{13}$$

The iterative procedure for the adaptation of the parameters and for the minimization of the criterion function can be summarized in Fig. 5. At first, design the center values of the membership function by experience, then get the weight vectors, set the learning rates, and set the tolerant rates. After that, read the sensor information and the desired output accelerations. Then, get the output by fuzzy logic algorithm (including fuzzification, fuzzy inference, and defuzzification; the three steps of defuzzification). And then, modify the parameters. Finally, evaluate the criterion function. If it is suitable to the criterion, the procedure is stopped, otherwise, repeat from reading sensor information to evaluating the criterion.

E. Algorithm to Suppress Redundant Rules

Forty-eight rules are defined in the fuzzy logic-based model. However, it is difficult to define the controller rules accurately and without redundant rules, if the number of input variables increases, or the number of the terms of variables increases to fit a more complex environment. To solve the problem, a selection algorithm is added to the fuzzy controller model to suppress redundant fuzzy rules automatically.

BEGIN

set m_{ij}, n_{ls} and σ_{ij} by experience;
get the weight vectors $w_{i,k}$ and $v_{k,l}$ from m_{ij} and n_{ls};
set learning rates ε_m, ε_n;
set tolerance δ;
DO (Training procedure)
Data input $\{u_1, u_2, u_3, u_4, u_5\}$ and $\{\hat{y}_1, \hat{y}_2\}$;
Fuzzification;
Fuzzy inference;
Defuzzification;
Data output $\{y_1, y_2\}$;
Adaptation of parameters m_{ij};
Adaptation of parameters n_{ls};
Evaluation of the criterion function E;
UNTIL ($E < \delta$) ;
END

Fig. 5. Algorithm to tune the parameters.

After the training to tune the model parameters, the learning algorithm to suppress redundant rules is considered. It is clear from Fig. 2 that the variables $w_{i,k}$ and $v_{k,l}$ that can be obtained from the model parameters m_{ij} and n_{ls} described earlier, determine the response of the fuzzy rules to the input signals. Every rule is related to a weight vector

$$W_k = \{w_{1,k}, \ldots, w_{5,k}, v_{k,1}, v_{k,2}\}, \quad k = 1, 2, \ldots \tag{14}$$

If the Euclidean distance between two weight vectors is small enough, both vectors will generate similar rules in the sense that a similar result is obtained for the same input. So, by calculating the Euclidean distances between the weight vectors, the redundant rules can be reduced. Based on this idea, the proposed algorithm is summarized in Fig. 6. At first, design the center values of the membership function by experience, then get the weight vectors, normalize the weights, and set the tolerant rates. After that, compare the Euclidean distance of every two vectors. If the distance is less than the tolerant value, remove one of the rules, which relate to the two vectors.

After applying the algorithm, a minimum number of rules are obtained. Thus, the minimum number of nodes in the second layer of the structure in Fig. 2 is obtained. For example, if the environment is simple, the tolerance can be chosen to be relatively large. Consequently, some of the rules will be suppressed and the number of useful rules will be smaller than 48. This algorithm has obvious benefits over rule bases with hundreds or even thousands of fuzzy rules.

```
BEGIN
    set m_ij, n_ls and σ_ij by experience;
    get the weight vectors w_i,k and v_k,l from m_ij and n_ls;
    normalize the values of weights to [0, 1];
    assign the number of rules to variable N;
    set tolerance δ;
    initialize index N1 = 1; N2 = 1;
    WHILE ( N1 < N ) DO
        N2 = N1 + 1;
        WHILE ( N2 < N + 1) DO
            d_N1N2 = ||W_N1 - W_N2||; //distance;
            IF ( d_N1N2 < δ ) THEN
                N = N - 1; //decrease number of rules;
                FOR ( N3 = N2 to N ) DO
                    W_N3 = W_N3+1; //reduce the vector W_N2;
                ENDDO
            ENDIF
            N2 = N2 + 1;
        ENDDO
        N1 = N1 + 1;
    ENDDO
END
```

Fig. 6. Algorithm to suppress redundant rules.

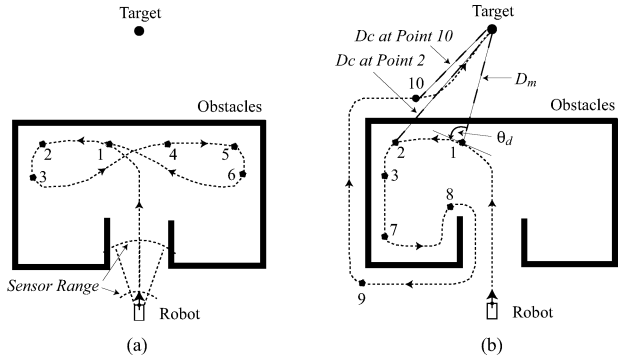

Fig. 7. Robot navigation in a "dead cycle" situation because of the limited sensor range. (a) Without the proposed state memory strategy. (b) With the proposed state memory strategy.

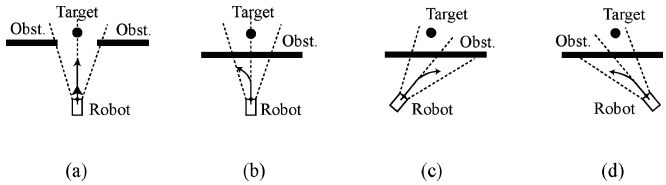

Fig. 8. Illustration of robot states. (a) and (b) State 0. (c) State 1. (d) State 2.

F. State Memorizing Strategy

Usually, the fuzzy behavior-based robot, like other reactive robots, suffers from the "symmetric indecision" and the "dead cycle" problems, such as in [6], [14], [23] and [24]. The proposed model resolves the "symmetric indecision" problem by designing several mandatory-turn rules (e.g., Rules 21, 22, 45, and 46). When an obstacle is near the front of the robot, and no obstacles exist on either side or the same situated obstacles exist on either side, the robot will turn left (or right) without hesitation. But, the "dead cycle" problem may still occur in some cases, even if the barrier following behavior has been added to the fuzzy inference rule base. For example, a robot wanders indefinitely in a loop inside a U-shaped obstacle, because it does not remember the place visited before, and its navigation is only based on the local sensed environment.

A typical "dead cycle" situation is shown in Fig. 7(a). First, the robot moves directly toward the target according to Rules 3 and 4, using the target seeking behavior, because there are no obstacles sensed in front of the robot and the robot thinks this is the ideal shortest path to the target. When the robot detects obstacles directly in front, it makes a left turn according to Rules 21 and 22, using the barrier following behavior. After that, the robot goes to the left along the obstacles according to Rules 11, 12, 17, and 18, using the barrier following behavior, because the obstacles are on the right side of the robot and the target is behind the obstacles. As a result, the robot moves along the curved path from Position 1 to 3 via 2. At Position 3, the robot turns to the left according to the barrier following behavior, since the obstacles exist on the right side and in the front side of the robot while the left side of the robot is empty and the target is on the left behind the robot. After Position 3, the robot is attracted to the target according to the target seeking behavior because there are either no obstacles or far away obstacles detected by the robot. Similarly, the robot follows the path from Position 4 to 6 via 5, and then to Position 1. A "dead cycle" occurs.

By careful examination of the earlier-mentioned "dead cycle" path under the control of the fuzzy rules, it can be found that Positions 3 and 6 are the critical points resulting in the "dead cycle" path. At Position 3, if the robot can go straight instead of a left turn, the problem may be resolved. Without changing the designed control rule base, if the robot can assume the target just on the right front side of the robot and behind the obstacles at Position 3, the robot will go straight and try to pass around the obstacles. Based on this idea, an assistant state memorizing strategy is developed for the system. There are three states designed for the robot shown in Fig. 8. State 0 represents the normal state or other situations except State 1 and 2; State 1 for both the target and obstacles being on the left side of the robot; and State 2 for both the target and obstacles being on the right side of the robot.

The flow diagram of the algorithm is shown in Fig. 9. Initially, the robot is in State 0, which means it is following the control rules. The robot changes to State 2 when the target and the obstacles are on the right side and the target direction θ_d shown in Fig. 7(b), which is the angle between the robot moving direction and the line connecting the robot center with the target angle, is increasing at Position 1, while the robot changes to State 1 if at Position 4. When the robot changes to State 1 or 2, the robot memorizes the state and the distance between the target and the robot denoted by D_m shown in Fig. 7(b). During State 1 or 2, the robot assumes the target just on the front left or right side of

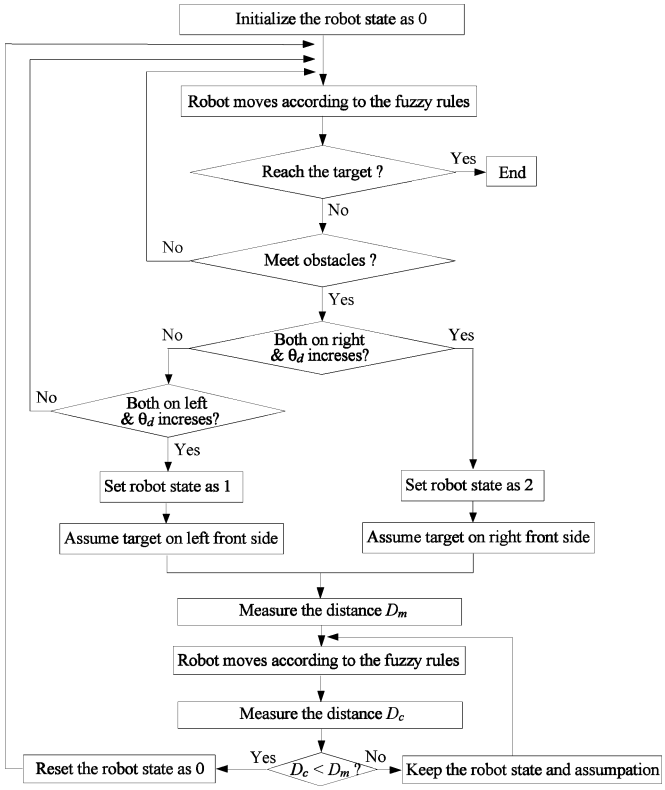

Fig. 9. Flow diagram of the state memorizing strategy.

the robot and behind the obstacles. Under this assumption and when the distance between the target and the current position D_c at Positions 1, 2, 3, 7, 8 and 9 being longer than D_m memorized at Position 1, the robot goes along the curve through Positions 1, 2, 3, 7, 8 and 9 in Fig. 7(b). The robot changes back to State 0 when D_c at Position 10, shown in Fig. 7(b), is shorter than D_m. This means that the robot has passed round the obstacles. Finally, the robot will reach the target and stop there.

III. Simulation Studies

To demonstrate the effectiveness of the proposed fuzzy logic-based controller, simulations using a mobile robot simulator (MobotSim Version 1.0.03 by Gonzalo Rodriquez Mir) are performed. The robot is designed as shown in Fig. 1. The diameter of the robot plate is set to 0.25 m, distance between wheels is set as 0.18 m, wheel diameter is 0.07 m, and wheel width is 0.02 m. In addition to the target sensor and the speed odometer, there are nine ultrasonic sensors mounted on the front part of the robot. The angle between sensors is 20°. The sensor ring radius is 0.1 m. The radiation cone of the sensors is 25°. The sensing range of the ultrasonic sensors is from 0.04 to 2.55 m. The upper bound of the wheel speed is 0.15 m/s. In every case, the environment is assumed to be completely unknown for the robot, except the target location; and the sensing range of the onboard robot sensors are limited.

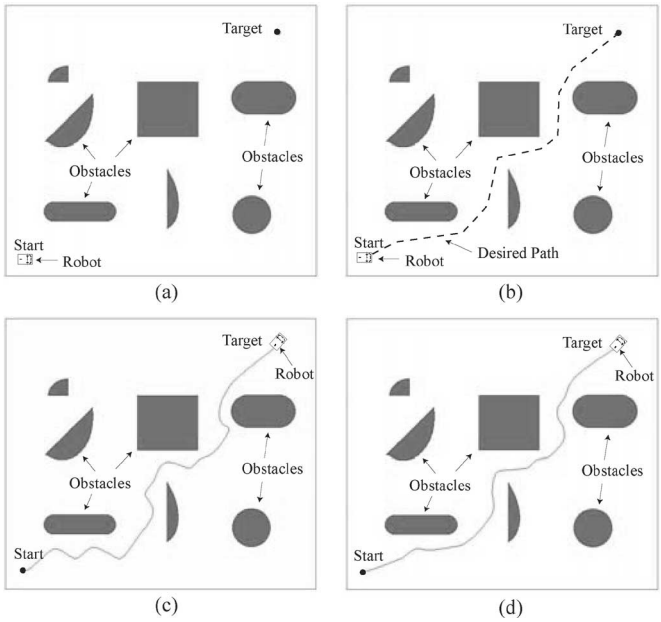

Fig. 10. Process of the training phrase. Process of the training phrase. (a) The workspace. (b) The desired trajectory. (c) The trajectory using initial model parameters and during the training. (d) The trajectory using new model parameters obtained from the training.

A. Offline Training Phase to Tune the Model Parameters

Initially in the training phase, the robot moves under the control of the supervisor. The goal of the learning step is to adjust the parameters of the membership functions and smooth the trajectory. To suit any situation for the mobile robot, the training environment should be designed relatively complicated, where there are left turns, right turns, straight stretches, and different obstacles for the robot. The workspace with a mobile robot, a target, and several obstacles is designed as in Fig. 10(a).

According to the experience, the model parameters are initialized as

$$\{m_{11}, m_{12}, m_{21}, m_{22}, m_{31}, m_{32}, m_{41}, m_{42}, m_{43}, m_{51}, m_{52}\} = \{100, 20, 100, 20, 100, 20, -45, 0, 45, 2, 10\} \quad (15)$$

$$\{n_{11}, n_{12}, n_{13}, n_{14}, n_{15}, n_{21}, n_{22}, n_{23}, n_{24}, n_{25}\} = \{10, 5, 0, -5, -10, 10, 5, 0, -5, -10\} \quad (16)$$

$$\{\sigma_{11}, \sigma_{12}, \sigma_{21}, \sigma_{22}, \sigma_{31}, \sigma_{32}, \sigma_{41}, \sigma_{42}, \sigma_{43}, \sigma_{51}, \sigma_{52}\} = \{160, 160, 160, 160, 160, 160, 90, 90, 90, 16, 16\}. \quad (17)$$

To get the desired outputs $[\hat{y}_1, \hat{y}_2]$, a reasonable path is drawn by an expert first as shown in Fig. 10(b). Then, the robot follows the path from the start position to the target position. The accelerations of the robot are then recorded at every time interval as the desired accelerations in the learning algorithm. To simplify the calculation, the learning rates ε_m and ε_n are selected as 0.01. During the training phase, the adaptation is done at every time interval. The trajectory during the training phase is shown in Fig. 10(c), where the trajectory is winding at the beginning period, but smoother at the end period. As mentioned

earlier, the effect of the controller outputs mainly depends on the center of the membership functions, while the widths can be disregarded. The parameters are tuned and set as in (7). In this simulation, after the learning, the better parameters of the membership functions are obtained as

$$\{m_{11}, m_{12}, m_{21}, m_{22}, m_{31}, m_{32}, m_{41}, m_{42}, m_{43}, m_{51}, m_{52}\} = \{94, 18, 105, 22, 95, 19, -48, 0, 47, 1.2, 9.4\} \quad (18)$$

$$\{n_{11}, n_{12}, n_{13}, n_{14}, n_{15}, n_{21}, n_{22}, n_{23}, n_{24}, n_{25}\} = \{9.5, 5.4, 0, -5.5, -9.3, 9.4, 5.6, 0, -5.3, -9.4\} \quad (19)$$

$$\{\sigma_{11}, \sigma_{12}, \sigma_{21}, \sigma_{22}, \sigma_{31}, \sigma_{32}, \sigma_{41}, \sigma_{42}, \sigma_{43}, \sigma_{51}, \sigma_{52}\} = \{160, 160, 160, 160, 160, 160, 90, 90, 90, 16, 16\}. \quad (20)$$

By using the new parameters, the mobile robot can work very well. The trajectory is shown in Fig. 10(d).

B. Offline Training Phase to Suppress Redundant Rules

As mentioned earlier, every rule is related to a weight vector in (14). And the weight vectors depend on (18) and (19). Table II shows the weights related to the rules.

After applying the selection algorithm, the number of the rules will be less than 48, depending on the tolerance δ. Table III shows the relationship between the number of the useful rules and the tolerance δ. The bigger δ, the less the number of rules, but the poorer the performance.

The robot trajectories are shown in Fig. 11 when $\delta = 0$ (a), 0.001 (b), 0.005 (c), and 0.01 (d). It is obvious that the trajectories in Fig. 11(a)–(c) are almost the same, except in Fig. 11(d). This means that, with only 38 rules, the system can get a reasonable result in making the robot navigate. There are 10 redundant rules that the system automatically removes. They are marked in Table I with a star. According to the experience, the redundant rules are Rules 12, 14, 16, 18, 20, 22, 34, 40, 44, and 46 in Table I. This can be explained from the rule base directly. For example, Rule 40 is similar to Rule 39, and is, therefore, redundant.

C. Effectiveness of the State Memory Strategy

After suppressing the redundant rules, the proposed controller is applied to the typical complicated situation that causes the "dead cycle" problems existing in many conventional approaches.

A robot moving in a U-shaped obstacles situation is shown in Fig. 12. Because of the limited sensors, the robot will get in the U-shaped area first, but by the state memory strategy, the robot will escape from the trap and eventually reache the target. It shows that the robot trajectory from the start position to the target does not suffer from the "dead cycle" problem.

D. More Complicated Static Environments

Robot navigation in a static complicated environment is demonstrated in Fig. 13, where the robot meets different situations that cause the "dead cycle" problem. In the first situation, a long wall is in front of the robot and the target is located behind the long wall. In the second situation, the robot is in an U-shaped area and the target is located left behind the obstacles. In the third situation, the robot is in an U-shaped area and the target is located right behind the obstacles. All of these situations would cause problems. For example, in Fig. 13, Positions 1, 3, 4, 5, 6, and 7 are the places that would cause the "dead cycle" problem. But, using the proposed fuzzy logic-based controller with the proposed state memorizing strategy, the robot can follow, pass round, and leave the obstacles, and finally reach the target along

TABLE II
WEIGHTS RELATED TO THE RULES AFTER THE TUNING

Rule No.	w_{1k}	w_{2k}	w_{3k}	w_{4k}	w_{5k}	v_{k1}	v_{k2}
1	94	105	95	-48	1.2	5.4	9.4
2	94	105	95	-48	9.4	-5.5	0
3	94	105	95	0	1.2	9.5	9.4
4	94	105	95	0	9.4	0	0
5	94	105	95	47	1.2	9.5	5.6
6	94	105	95	47	9.4	0	-5.3
7	94	105	19	-48	1.2	0	5.6
8	94	105	19	-48	9.4	-5.5	0
9	94	105	19	0	1.2	0	5.6
10	94	105	19	0	9.4	-5.5	0
11	94	105	19	47	1.2	-5.5	0
12	94	105	19	47	9.4	-9.3	-5.3
13	94	22	19	-48	1.2	-5.5	0
14	94	22	19	-48	9.4	-9.3	-5.3
15	94	22	19	0	1.2	-5.5	0
16	94	22	19	0	9.4	-9.3	-5.3
17	94	22	19	47	1.2	-5.5	0
18	94	22	19	47	9.4	-9.3	-5.3
19	18	22	19	-48	1.2	-5.5	0
20	18	22	19	-48	9.4	-9.3	-5.3
21	18	22	19	0	1.2	-5.5	0
22	18	22	19	0	9.4	-9.3	-5.3
23	18	22	19	47	1.2	0	-5.3
24	18	22	19	47	9.4	-5.5	-9.4
25	18	105	19	-48	1.2	0	0
26	18	105	19	-48	9.4	-5.5	-5.3
27	18	105	19	0	1.2	5.4	5.6
28	18	105	19	0	9.4	-5.5	-5.3
29	18	105	19	47	1.2	0	0
30	18	105	19	47	9.4	-5.5	-5.3
31	18	105	95	-48	1.2	0	-5.3
32	18	105	95	-48	9.4	-5.5	-9.4
33	18	105	95	0	1.2	5.4	0
34	18	105	95	0	9.4	0	-5.3
35	18	105	95	47	1.2	5.4	0
36	18	105	95	47	9.4	0	-5.3
37	18	22	95	-48	1.2	0	-5.3
38	18	22	95	-48	9.4	-5.5	-9.4
39	18	22	95	0	1.2	0	-9.4
40	18	22	95	0	9.4	-5.5	-9.4
41	18	22	95	47	1.2	0	-5.3
42	18	22	95	47	9.4	-5.5	-9.4
43	94	22	95	-48	1.2	-5.5	0
44	94	22	95	-48	9.4	-9.3	-5.3
45	94	22	95	0	1.2	-5.5	0
46	94	22	95	0	9.4	-9.3	-5.3
47	94	22	95	47	1.2	0	-5.3
48	94	22	95	47	9.4	-5.5	-9.4

TABLE III
RELATIONSHIP BETWEEN THE NUMBER OF THE USEFUL RULES AND THE TOLERANCE δ

tolerance δ	0	0.0005	0.001	0.005	0.01	0.05
$numberofrules$	48	48	47	38	26	18

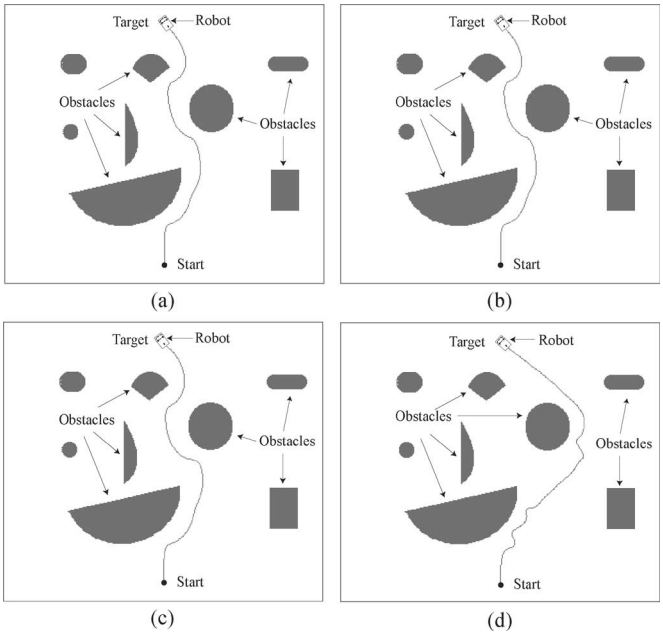

Fig. 11. Mobile robot trajectories with different number of rules when the tolerance δ is selected as (a) 0 with 48 rules. (b) 0.001 with 47 rules. (c) 0.005 with 38 rules. (d) 0.01 with 26 rules.

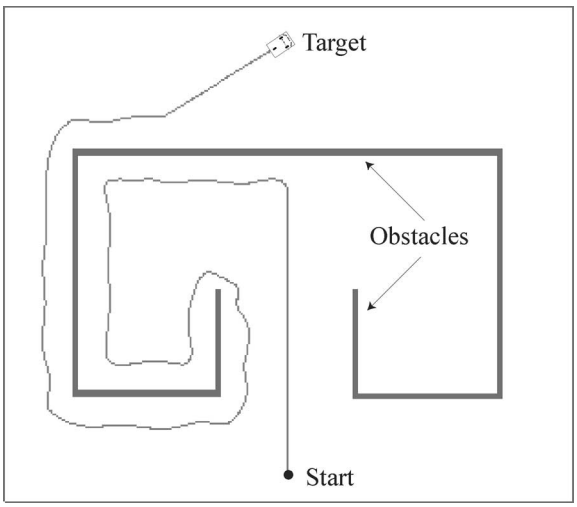

Fig. 12. Robot navigation in a complicated situation without suffering from the "dead cycle" problems.

reasonable trajectories without suffering from the "dead cycle" problem.

In Fig. 13, initially the robot sees no obstacles in its direction toward the target. Thus, the robot assumes all free space in its front and moves straight toward the target from the start point. However, when the robot arrives at Position 1, it senses the obstacles in its front, near its left side, and far from its right side. According to the designed fuzzy control rules and the proposed state memory strategy, the robot changes its state from 0 to 1, memorizes the distance between Position 1 and the target, and turns right to avoid the obstacles. When the robot arrives at Position 2, there are no obstacles between the robot and the target, and the distance between the robot and the target is shorter than the memorized distance at Position 1. Thus, the robot backs

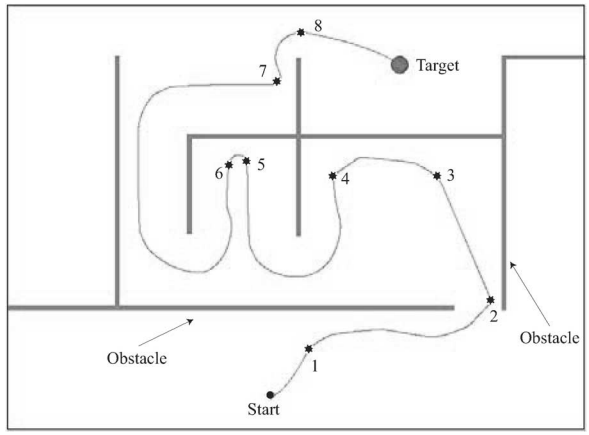

Fig. 13. Robot navigation in a static complex case.

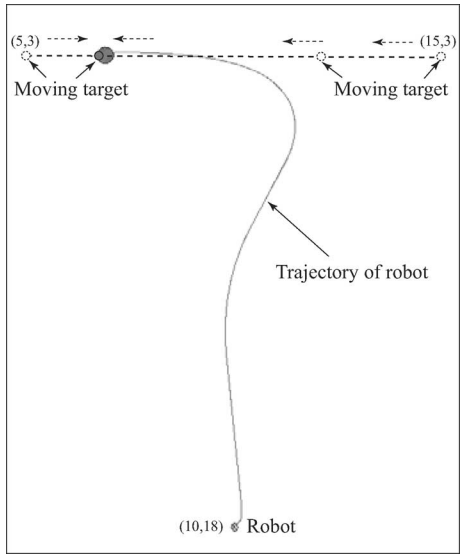

Fig. 14. Robot navigation in a dynamic environment with target moving in a line.

to state 0 and goes straight toward the target again. However, when the robot arrives at Position 3, it senses the obstacles in its front, near its right side, and far from its left side. The robot changes its state from 0 to 2, memorizes the distance between Position 3 and the target, and turns left to avoid the obstacles. When the robot arrives at Position 4, even if the target is located on the left side of robot and there are no obstacles on the left side of the robot, because the distance between the robot and target is longer than the memorized distance, the robot stays in State 2 and follows the obstacles up to Position 9 via Position 5, 6, 7, and 8. The robot changes its state to 0 at Position 9, and then goes straight to the target.

E. Dynamic Environments and Velocity Analysis

Robot navigation in dynamic environments are conducted in this section. The first case shows the robot navigation in a target moving environment. Assume a target is moving in a line from the position (5,3) to (15,3) and then back to (5,3), while the robot starts from position (10,18) in the workspace and is moving to the right. Fig. 14 shows the trajectory of the robot.

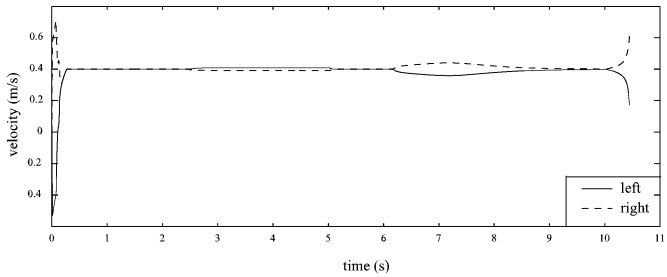

Fig. 15. Velocities of the robot navigation in a dynamic environment with target moving in a line.

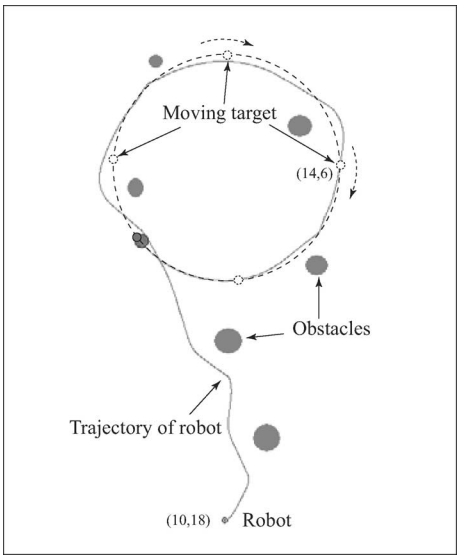

Fig. 16. Robot navigation in a dynamic environment with target moving in a cycle with obstacles.

At the beginning, the robot turns left, and then goes straight toward the target, then turns right, and then turns left following the target. A smooth trajectory is executed in traveling to the target.

The recorded velocity profile of both wheels in this simulation is presented in Fig. 15. It can be seen from this figure that when the robot turns left, the velocity of the right wheel increases much more than the velocity of the left wheel at the beginning. The velocities of both wheels are the same when the robot goes straight. The velocity of the right wheel increases a little and the velocity of the left wheel decreases a little when the robot turns left during its navigation procedure.

In the second case, assume that the target moves in a cycle in a workspace in which a few obstacles exist. A smooth trajectory is executed in obstacle avoidance and traveling to the target as shown in Fig. 16.

The recorded velocity profile of both wheels in this simulation is presented in Fig. 17. It can be seen from this figure that at the beginning as the robot turns left, the velocity of the right wheel increases much more than the velocity of the left wheel. The velocity of the left wheel increases a little and the velocity of the right wheel decreases a little when the robot turns right when there are no obstacles in front of it. When the robot encounters

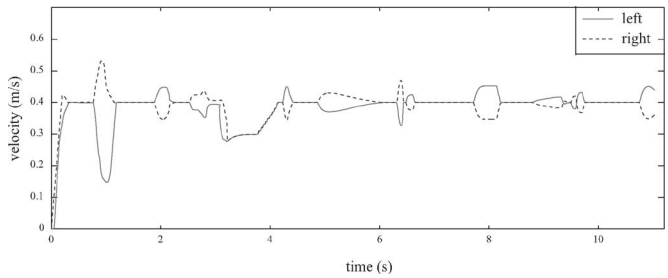

Fig. 17. Velocities of the robot navigation in a dynamic environment with target moving in a cycle with obstacles.

obstacles, it will turn left by increasing the velocity of the right wheel and decreasing the velocity of the left wheel, or turn right by increasing the velocity of the left wheel and decreasing the velocity of the right wheel.

IV. Conclusion

In this paper, a novel fuzzy logic-based control system, combining sensing and a state memory strategy, is proposed for real-time reactive navigation of a mobile robot. Under the control of the proposed fuzzy logic-based model, the mobile robot can autonomously reach the target along a smooth trajectory with obstacle avoidance. Several features of the proposed approach are summarized as follows.

1) The proposed model keeps the physical meanings of the variables and parameters during the processing, while conventional models [10] cannot.
2) Forty-eight fuzzy rules and two behaviors are designed in the proposed model, much fewer than conventional approaches that use hundreds of rules (e.g., 600 rules in [5]).
3) The structure of the proposed fuzzy logic-based model is very simple with only 11 nodes in the first layer of the structure, while hundreds of nodes are necessary in some conventional fuzzy logic-based models (e.g., 240 nodes needed in the first layer in [10]).
4) The proposed selection algorithm can automatically suppress redundant fuzzy rules when there is a need.
5) A very simple yet effective state memory strategy is designed. It can resolve the "dead cycle" problem existing in some previous approaches [6] without changing the fuzzy control rule base.

References

[1] E. Aguirre and A. Gonzalez, "A fuzzy perceptual model for ultrasound sensors applied to intelligent navigation of mobile robots," *Appl. Intell.*, vol. 19, pp. 171–187, 2003.
[2] C. Bruckhoff and P. Dahm, "Neural fields for local path planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. Innovations Theory Pract Appl.*, vol. 3, 1998, pp. 1431–1437.
[3] I. J. Cox, "Blanche: Position estimation for an autonomous robot vehicle," in *Proc. IEEE/RSJ Int. Workshop Robot. Syst.*, 1989, pp. 432–439.
[4] P. Dahm, C. Bruckhoff, and F. Joublin, "A neural field approach for robot motion control," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, vol. 4, 1998, pp. 3460–3466.
[5] J. Godjevac, "A learning procedure for a fuzzy system: Application to obstacle avoidance," in *Proc. Int. Symp. Fuzzy Logic*, Zurich, Switzerland, 1995, pp. 142–148.
[6] J. Godjevac and N. Steele, "Neuro-fuzzy control of a mobile robot," *Neurocomputing*, vol. 28, pp. 127–143, 1999.

[7] H. Hagras, V. Callaghan, and M. Colley, "Online learning of the sensors' fuzzy membership functions in autonomous mobile robots," in *Proc. 2000 IEEE Int. Conf. Robot. Autom.*, San Francisco, CA, Apr., vol. 4, pp. 3233–3238.
[8] P. S. Lee and L. L. Wang, "Collision avoidance by fuzzy logic control for automated guided vehicle navigation," *J. Robot. Syst.*, vol. 11, no. 8, pp. 743–760, 1994.
[9] H. Li and S. X. Yang, "A behavior-based mobile robot with a visual landmark recognition system," *IEEE Trans. Mechatronics*, vol. 8, no. 3, pp. 390–400, Sep. 2003.
[10] G. N. Marichal, L. Acosta, L. Moreno, J. A. Méndez, J. J. Rodrigo, and M. Sigut, "Obstacle avoidance for a mobile robot: A neuro-fuzzy approach," *Fuzzy Sets Syst.*, vol. 124, no. 2, pp. 171–179, Dec. 2001.
[11] H. Noborio, R. Nogami, and S. Hirao, "A new sensor-based path-planning algorithm whose path length is shorter on the average," in *Proc. IEEE Int. Conf. Robot. Autom.*, New Orleans, LA, Apr. 2004, vol. 5, pp. 2832–2839.
[12] R. Nogami, S. Hirao, and H. Noborio, "On the average path lengths of typical sensor-based path-planning algorithms by uncertain random mazes," in *Proc. IEEE Int. Symp. Comput. Intell. Robot. Autom.*, Kobe, Japan, Jul. 16–20, 2003, vol. 2, pp. 471–478.
[13] J. Rosenblatt, "The distributed architecture for mobile navigation," *J. Exp. Theor. Artif. Intell.*, vol. 9, no. 2–3, pp. 339–360, Apr.–Sep. 1997.
[14] P. Rusu, E. M. Petriu, T. E. Whalen, A. Cornell, and H. J. W. Spoelder, "Behavior-based neuro-fuzzy controller for mobile robot navigation," *IEEE Trans. Instrum. Meas.*, vol. 52, no. 4, pp. 1335–1340, Aug. 2003.
[15] B. S. Ryu and H. S. Yang, "Integration of reactive behaviors and enhanced topological map for robust mobile robot navigation," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 29, no. 5, pp. 474–485, Sep. 1999.
[16] A. Saffiotti, "Fuzzy logic in autonomous robotics: Behavior coordination," in *Proc. 6th IEEE Int. Conf. Fuzzy Syst.*, Barcelona, Spain, Jul. 1997, vol. 3, pp. 573–578.
[17] A. Saffiotti, "The uses of fuzzy logic in autonomous robot navigation," *Soft Comput.*, vol. 1, pp. 180–197, 1997.
[18] A. Saffiotti, E. H. Ruspini, and K. Konolige, "Using fuzzy logic for mobile robot control," in *Practical Applications of Fuzzy Technologies*, H.-J. Zimmermann, Ed. Boston, MA: Kluwer Academic, 1999, pp. 185–206.
[19] A. Sankaranarayanan and M. Vidyasagar, "A new path planning algorithm for moving a point object amidst unknown obstacles in a plane," in *Proc. IEEE Int. Conf. Robot. Autom.*, Cincinnati, OH, May1990, vol. 3, pp. 1930–1936.
[20] K. T. Song and L. H. Sheen, "Heuristic fuzzy-neuro network and its application to reactive navigation of a mobile robot," *Fuzzy Sets Syst.*, vol. 110, no. 3, pp. 331–340, Mar. 2000.
[21] I. H. Suh, S. Lee, B. O. Kim, B. J. Yi, and S. R. Oh, "Design and implementation of a behavior-based control and learning architecture for mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, Taipei, Taiwan, Sep. 2003, vol. 1, pp. 4142–4147.
[22] P. Vadakkepat, O. C. Miin, X. Peng, and T. H. Lee, "Fuzzy behavior-based control of mobile robots," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 4, pp. 559–564, Aug. 2004.
[23] W. L. Xu and S. K. Tso, "Real-time self-reaction of a mobile robot in unstructured environments using fuzzy reasoning," *Eng. Appl. Artif. Intell.*, vol. 9, no. 5, pp. 475–485, 1996.
[24] W. L. Xu and S. K. Tso, "Sensor-based fuzzy reactive navigation of a mobile robot through local target switching," *IEEE Trans. Syst., Man Cybern., C, Appl. Rev.*, vol. 29, no. 3, pp. 451–459, Aug. 1999.
[25] S. X. Yang and M. Meng, "An efficient neural network method for real-time motion planning with safety consideration," *Robot. Auton. Syst.*, vol. 32, no. 2–3, pp. 115–128, 2000.
[26] S. X. Yang and M. Meng, "Neural network approaches to dynamic collision-free trajectory generation," *IEEE Trans. Syst., Man, Cybern., B, Cybern.*, vol. 31, no. 3, pp. 302–318, Jun. 2001.
[27] S. X. Yang and Q.-H. M. Meng, "Real-time collision-free motion planning of mobile robots using neural dynamics based approaches," *IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1541–1552, Nov. 2003.
[28] X. Yang, M. Moallem, and R. V. Patel, "A novel intelligent technique for mobile robot navigation," in *Proc. IEEE Conf. Control Appl.*, Jun. 2003, vol. 1, pp. 674–679.
[29] T. Yoshioka and H. Noborio, "On a proof of the deadlock-free property in an on-line path-planning by using world topology," in *Proc. IEEE/RSJ/GI Int. Conf. Intell. Robot. Syst. 1994 Adv. Robot. Syst. Real World*, Munich, Germany, Sep.1994, vol. 3, pp. 1946–1953.
[30] E. Zalama, J. Gmez, M. Paul, and J. R. Pern, "Adaptive behavior navigation of a mobile robot," *IEEE Trans. Syst., Man, Cybern., A, Syst. Humans*, vol. 32, no. 1, pp. 160–169, Jan. 2002.

Anmin Zhu received the B.Sc. and M.Sc. degrees in computer science from Tongji University, Shanghai, China, in 1987 and 1990, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Guelph, Guelph, ON, Canada, in 2005.

Currently, he is a Faculty Member at the College of Information Engineering, Shenzhen University, Shenzhen, China. His current research interests include fuzzy systems, neural networks, robotics, self-organizing arrangement, machine intelligence, and multiagent systems.

Simon X. Yang (S'97–M'99) received the B.Sc. degree in engineering physics from Peking University, Beijing, China, in 1987, the first of two M.Sc. degrees in biophysics from the Chinese Academy of Sciences, Beijing, in 1990, the second M.Sc. degree in electrical engineering from the University of Houston, Houston, TX, in 1996, and the Ph.D. degree in electrical and computer engineering from the University of Alberta, Edmonton, AB, Canada, in 1999. He serves on the Editorial Boards of the journal *Dynamics of Continuous, Discrete, and Impulse Systems*, the *International Journal of Information Acquisition*, and the *International Journal of Computational Intelligence and Applications*.

He joined the University of Guelph, Guelph, ON, Canada in 1999, where he is currently an Associate Professor and the Director of Advanced Robotics and Intelligent Systems Laboratory. His current research interests are in the areas of robotics, intelligent systems, sensors, control systems, image and signal processing, neurocomputation, and bioinformatics. He is the author or coauthor of over 250 refereed journal papers, book chapters, and conference papers.

Dr. Yang was the General Chair of the 2006 International Conference on Sensing, Computing, and Automation. He was the recipient of the 2004–2006 Presidential Distinguished Professor Award at the University of Guelph.