Chapter 12
An Adaptive Neuro-fuzzy Controller for Robot Navigation

Anmin Zhu and Simon X. Yang

Abstract Real-time autonomous navigation in unpredictable environments is an essential issue in robotics and artificial intelligence. In this chapter, an adaptive neuro-fuzzy controller is proposed for mobile robot navigation with local information. A combination of multiple sensors is used to sense the obstacles near the robot, the target location, and the current robot speed. A fuzzy logic system with 48 fuzzy rules is designed. Two learning algorithms are developed to tune the parameters of the membership functions in the proposed neuro-fuzzy model and automatically suppress redundant fuzzy rules from the rule base. The "dead cycle" problem is resolved by a state memory strategy. Under the control of the proposed neuro-fuzzy model, the mobile robot can preferably "see" the surrounding environment, avoid static and moving obstacles automatically, and generate reasonable trajectories toward the target. The effectiveness and efficiency of the proposed approach are demonstrated by simulation and experiment studies.

12.1 Introduction

Mobile robot navigation using only onboard sensors is an essential issue in robotics and artificial intelligence. For real-time autonomous navigation in unknown and changing environments, 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

Anmin Zhu
The College of Information Engineering, Shenzhen University, Shenzhen 518060, P.R.China, e-mail: azhu@szu.edu.cn

Simon X. Yang
The Advanced Robotics and Intelligent System (ARIS) Laboratory, School of Engineering, University of Guelph, Guelph, ON N1G 2W1, Canada, e-mail: syang@uoguelph.ca

obstacle avoidance, and controlling the robot direction and velocity to reach the target.

There are some conventional approaches to real-time autonomous navigation. Map-based methods [3,8] combine a graph searching technique with obstacle pruning to generate trajectory for mobile robots. These graph-based methods first need to construct a data structure that is then used to find paths between configurations of the robot. The data structure tends to be very large, and the geometric search computation required is complex and expensive. In addition, these methods are usually used only for robot path planning without considering the robot motion control. Artificial potential field methods [4, 7] assumes that each obstacle in the environment exerts a repulsive force on the mobile robot, and the target exerts an attractive force. These two types of forces are combined at every step and used to determine the next step of the robot movement. These methods are faster than map-based methods, but have to consider robot as a point, being trapped at local minima; unintended stoppage between closely spaced obstacles; and oscillations in the presence of multiple obstacles or in narrow passages. In order to overcome shortcomings of the potential field methods, the vector field method was proposed and extended [6, 19]. These methods are faster then potential field and less trapped in local minima. However the methods ignored the dynamic and kinematic constraints of mobile robots, And may fail to choose the most appropriate direction and get trapped in a "dead cycle" situation. Neural network-based approaches [12,21] use a neural field approach described by an integro-differential equation. It 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. However, the robot path is not continuous and the kinematic constraint of the robot is not considered in these algorithms.

Because of the ability of making inference under uncertainty, fuzzy logic approaches are proposed for controlling a mobile robot in dynamic unknown environments. 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. Yang and Patel [22] 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 sub-problem and obstacle avoidance sub-problem are solved by two separate fuzzy logic systems. But this algorithm focus on the direction control without considering velocity control. Furthermore, it cannot solve the "dead cycle" problem in environments with an U-shaped obstacle. 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, but the model focuses on map building, and thus is computationally expensive. Park and Zhang [15] proposed a dual fuzzy logic approach. But the design of the two 81 fuzzy rules is not systematized, and redundance is obvious. 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, and more importantly it lacks self-organizing and self-tuning mechanisms. This makes it difficult to decide the parameters of the membership functions. Another drawback is 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 model-free systems which are organized in a way to simulate the cells of a human brain. They learn from the underlying relationships of data. Neural networks have a self-learning capability, self-tuning capability, dot no need to know data relationships, 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. Marichal *et al*. [11] presented a neuro-fuzzy 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 to very simple environments. Song *et al*. [17] 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 is, the more difficult to construct the patterns are. Hagras *et al*. [5] 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 and Petriu [16] proposed a neuro-fuzzy controller for mobile robot navigation in an indoor environment. Infrared and contact sensors are used for detecting targets and avoiding collisions. Two levels with several behaviors are designed for the controller. Fuzzy inference is used in each 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 neuro-fuzzy 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 neural network. Furthermore, there is no evidence that the controller is able to resolve the "dead cycle" problem. Some methods have been proposed to resolve the "dead cycle" problem, such as [13, 14], However, 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; this is difficult for real robot systems to realize. Besides the robot navigation, adaptive neuro-fuzzy based approaches have been applied to wide applications, such as robotic manipulator control [18], harmonic-drive robotic actuators [10], and simultaneous localization and mapping of mobile either robots or vehicles [2].

The advantages and disadvantages of previous models for the motion planning and control of a mobile robot using only onboard sensors are summarized in Table 12.1. The theoretical design, suitable to complicated environments, and "dead cycle" problem are the main problems of these existing approaches.

In this chapter, a novel adaptive neuro-fuzzy controller is presented for the reactive navigation of mobile robots in unknown environments. The work presented in this chapter is the summary and extension of our previous work [23, 24]. The inputs of the controller are the environment information around the robot, 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 adaptive neuro-fuzzy controller, the mobile robot can generate reasonable trajectories toward the target in various situations.

This chapter is organized as follows. Section 12.2 presents the overall structure of the proposed adaptive neuro-fuzzy controller to mobile robot navigation with limited sensors. The adaptive neuro-fuzzy controller is designed in Section 12.3. Sections 12.3.5 and 12.3.6 describe the algorithms to tune the model parameters, and to suppress redundant rules. The state memorizing strategy is given in Section 12.3.7. Sections 12.4 and 12.5 provide simulation and experiment results. Finally some concluding remarks are given in Section 12.6.

Table 12.1 Summary of the previous methods for mobile robot navigation

Methods (references)	Main advantages	Main disadvantages
Map-based [8]; [3]	Ideal simple and visual	Computation expensive, Not for real-time control
Potential field [7]; [4]	Faster than graph-based	Local minima, "Dead cycle" problem Consider robot to a point
VFH [19]; [6]	Faster than potential field Less trapped in local minima	Ignored dynamic constraints, Ignored dynamic constraints, "Dead cycle" problem
Neural networks-based [21]; [12]	Faster than potential field Without local minima	Discrete paths, Consider constant speed,
Fuzzy logic [22]; [15]	Various environments	No systematic design, Speed jump, "Dead cycle" problem
Neuro-fuzzy [16]; [11]	Various environments Learning capability	Difficult to design, Complicated, "Dead cycle" problem

12.2 The Overall Structure of the Neuro-fuzzy Controller

Navigation is a very easy task for human beings or animals. Like the human thinking process, while a mobile robot is moving in an unknown and changing environment, it is important to the compromise between avoiding collisions with obstacles and moving toward targets, depending on the sensed information about the environment. Without any obstacles, the robot moves straight toward the target, accelerating or decelerating according to the target direction and the distance between the robot and the target. When there are obstacles in the environment, the robot moves according to the distance to the obstacles and the location of the target. When obstacles are very close, the robot should slow down and make a turn to avoid the obstacles. The first part of a robot system are the sensor systems. Sensors must be mounted on the robot to sense the environment and interpret the sensed information. The main sensors of the mobile robot is shown in Figure 12.1. The robot has two front co-axle wheels driven by different motors separately, and a third passive omnidirectional caster. Through 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, a global positioning system (GPS), or a indoor positioning system [20] would be used to obtain the target direction. A speed meter is equipped on the robot to measure the current robot speed.

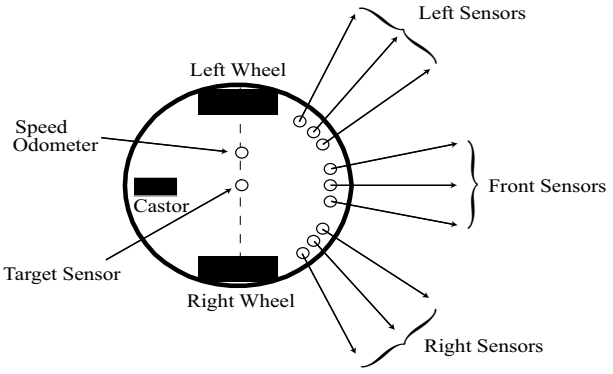

Fig. 12.1 The mobile robot model with onboard sensors

The robot motion is controlled by adjusting the speeds or accelerations of two driven wheels. At first, the robot system has to judge the distance as "far" or "close", the speed as "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 models. However, they are easy for human beings to use, as people do not need precise, numerical information input to make a decision, and they are able to perform highly adaptive control regardless, because there is "fuzzy" concept in human knowledge.

"Fuzzy" concept of human knowledge can be used in logic systems. The logic system then becomes a fuzzy logic system, which is basically an expert system based on a conventional logic system, but the conditions and results in the "IF THEN" rules are described in fuzzy terms. 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 with unclearly expressed concepts, to deal with imprecise or imperfect information, and to improve knowledge representation and uncertain reasoning. Therefore, a fuzzy logic method is selected to deal with the sensor-based robot motion control problem. The proposed fuzzy logic method is simple, easy to understand, having the intelligence of human beings, and having a quick reaction capability.

Fuzzy logic offers a framework for representing imprecise, uncertain knowledge. But the disadvantages are that fuzzy logic needs highly abstract heuristics, needs experts for rule discovery with data relationships, and more importantly it lacks self-organizing and self-tuning mechanisms. This makes it difficult to decide the parameters of the membership functions. Another drawback is lack of a systematic procedure to transform expert knowledge into the rule base. This results in many redundant rules in the rule base. Neural networks are unmodel-based systems which are organized in a way to simulate the cells of a human brain. They learn from the underlying relationships of data. Neural networks have a self-learning capability, a self-tuning capability, do not need to know data relationships, 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 per-

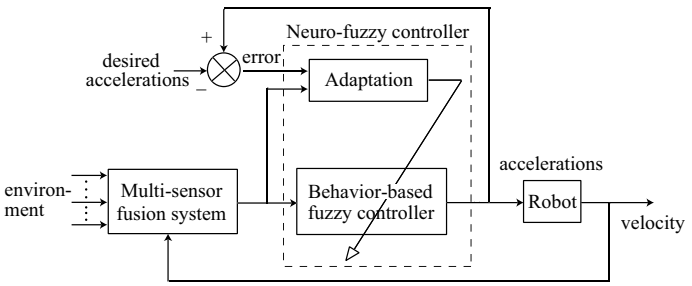

Fig. 12.2 A schematic diagram of the proposed adaptive neuro-fuzzy controller

formance. Figure 12.2 shows a brief structure of the proposed adaptive neuro-fuzzy controller, which consists of a fuzzy controller and a learning adaptation model, and the connection with other parts of the robot system, which consists of a multi-sensor fusion system and a robot acceleration control system.

12.3 Design of the Neuro-fuzzy Controller

The structure of the proposed adaptive neuro-fuzzy controller is shown in Figure 12.3. This is a five-layer neuro-fuzzy network. The inputs of the fuzzy controller are the outputs from the multi-sensor 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 neuro-fuzzy methods (e.g., [5, 11, 16, 17]). First of all, far fewer rules are needed in the proposed approach, while some conventional methods need a large number of rules. This simplifies the structure of the neuro-fuzzy model by reducing the number of the hidden layer neuron, and reduces the computational time. The physical meaning of the parameters remains the same during the training, which is lost in conventional neuro-fuzzy methods. The neural network based methods are developed to improve the performance in the proposed approach, including the parameter tuning and the redundant rule reducing. Lastly, a state memorizing strategy is designed to resolve the "dead cycle" problem in the proposed approach.

Fuzzification, inference mechanism, and defuzzification are considered to create the fuzzy logic controller. After that, neural network approaches are used to improve the performance.

12.3.1 Fuzzification and Membership Function Design

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, non-fuzzy input data is mapped to fuzzy sets by treating them as Gaussian, triangle, trapezoid, sharp peak membership functions, etc. In this chapter, triangle functions, S-type and Z-type functions are chosen to represent fuzzy membership functions. Because the number of the potential rules of fuzzy logic systems depends on the number of linguistic terms of the input variables, a relatively small number of terms for the input variables are selected. The input variables d_l, d_f and d_r are simply expressed

using two linguistic terms: NEAR and FAR. The variable θ_d is expressed by three terms: LEFT, CENTER and RIGHT. The variable r_s is expressed by two terms: FAST and SLOW. The output variables a_l and a_r are expressed by five terms: PB, PS, Z, NS, and NB abbreviated from positive big, positive small, zero, negative small and negative big respectively, and denote a big increment, a small increment, no change, a small decrement, and a big decrement of accelerations of two wheels. It is noted that the real speeds of the two wheels are constrained in the desired upper bounds. The robot will stop whenever it approaches the target.

The membership functions for all the terms of the input and output variables in this controller are shown in Figure 12.4. The outputs of the fuzzification procedure are given as follows. For a triangle function,

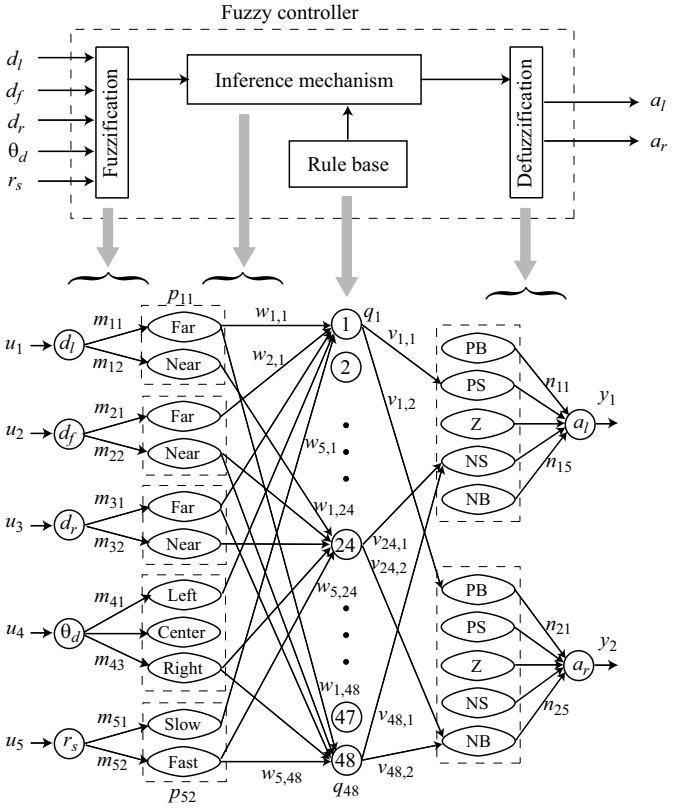

Fig. 12.3 Block diagram of the proposed neuro-fuzzy controller, d_l, d_f, d_r: obstacle distances to the left, front and right of the robot; θ_d: target direction; r_s: current robot speed; a_l, a_r: accelerations of the left and right wheels; $[u_1,u_2,u_3,u_4,u_5]=[d_l,d_f,d_r,\theta_d,r_s]$; m_{ij}: the centers of membership functions 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}; and $[y_1,y_2]=[a_l,a_r]$

$$p_{ij} = \begin{cases} 1 - \frac{2|u_i - m_{ij}|}{\sigma_{ij}}, \text{ if } m_{ij} - \frac{\sigma_{ij}}{2} < u_i < m_{ij} + \frac{\sigma_{ij}}{2}, \\ 0, \qquad\qquad \text{otherwise;} \end{cases} \tag{12.1}$$

for a S-type function,

$$p_{ij} = \begin{cases} 0, & \text{if } u_i < m_{ij} - \frac{\sigma_{ij}}{2}, \\ 1, & \text{if } u_i > m_{ij}, \\ 1 - \frac{2|u_i - m_{ij}|}{\sigma_{ij}}, & \text{otherwise;} \end{cases} \tag{12.2}$$

for a Z-type function,

$$p_{ij} = \begin{cases} 0, & \text{if } u_i > m_{ij} + \frac{\sigma_{ij}}{2}, \\ 1, & \text{if } u_i < m_{ij}, \\ 1 - \frac{2|u_i - m_{ij}|}{\sigma_{ij}}, & \text{otherwise,} \end{cases} \tag{12.3}$$

where $i = 1, 2, \cdots, 5$, is the number of input signals; $j = 1, 2, \cdots, 5$, is the number of terms of the input variables; p_{ij} is the degree of membership for the i-th input corresponding to the j-th term of the input variable; u_i is the i-th 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 i-th input and the j-th term of the input variable; and σ_{ij} is the width of the membership function corresponding to the i-th input and the j-th term of the input variable. For example, u_4=θ_d, represents the input value about the target direction; $m_{42} = 0$ means the value of the membership function center related to the 2th term ("center") of the 4th input variable (u_4 or θ_d) is 0; and $\sigma_{42} = 90$ is the width of the membership function.

12.3.2 Inference Mechanism and Rule Base Design

The inference mechanism is responsible for decision-making in the fuzzy controller using approximate reasoning. The rule base is essential for the controller, and stores the rules governing the input and output relationship of the proposed controller. The inference rules are in same form as the example shown in Figure 12.5.

48 rules are formulated for the proposed controller given in Table 12.2. There are three behaviors in the rule base with TS representing the target seeking behavior, OA representing the obstacle avoidance behavior, and BF representing the barrier following 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. The barrier following behavior is used to follow the obstacle by moving along or slightly toward the obstacle when the target is behind the obstacle and the obstacle is not too far or too close to the robot.

Table 12.2 The rule base of the proposed fuzzy logic controller. N: near; F: far; L: left; C: center; R: righ; SL: slow; FS: fast; PB: positive big, PS: positive small, Z: zero, NS: negative small, NB: negative big; TS: target seeking; OA: obstacle avoidance; BF: barrier following; $*$: removed rules

Rule	Input					Output			Redundant
No.	d_l	d_f	d_r	θ_d	r_s	a_l	a_r	behavior	rule
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	BF	
12	F	F	N	R	FS	NB	NS	BF	$*$
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	BF	
16	F	N	N	C	FS	NB	NS	BF	$*$
17	F	N	N	R	SL	NS	Z	BF	
18	F	N	N	R	FS	NB	NS	BF	$*$
19	N	N	N	L	SL	NS	Z	BF	
20	N	N	N	L	FS	NB	NS	BF	$*$
21	N	N	N	C	SL	NS	Z	BF	
22	N	N	N	C	FS	NB	NS	BF	$*$
23	N	N	N	R	SL	Z	NS	BF	
24	N	N	N	R	FS	NS	NB	BF	
25	N	F	N	L	SL	Z	Z	BF	
26	N	F	N	L	FS	NS	NS	BF	
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	BF	
30	N	F	N	R	FS	NS	NS	BF	
31	N	F	F	L	SL	Z	NS	BF	
32	N	F	F	L	FS	NS	NB	BF	
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	BF	
38	N	N	F	L	FS	NS	NB	BF	
39	N	N	F	C	SL	Z	NB	BF	
40	N	N	F	C	FS	NS	NB	BF	$*$
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	BF	
46	F	N	F	C	FS	NB	NS	BF	$*$
47	F	N	F	R	SL	Z	NS	OA	
48	F	N	F	R	FS	NS	NB	OA	

In every rule, the IF part is defined alternatively: If condition A is true, AND B is true, AND C is true, AND $\cdots$ with five conditions. Using the fuzzy logic operators, the AND can mathematically be represented by the min operator in the aggregation step. The output of the aggregation procedure to get degree of IF part of every rule is given as

$$q_k = \min\{p_{1k_1}, p_{2k_2}, p_{3k_3}, p_{4k_4}, p_{5k_5}\}, \tag{12.4}$$

where q_k is the conjunction degree of the IF part of the k-th rule, $k = 1, 2, \cdots, 48$; and p_{ik_i} is the degree of the membership for the i-th input contributing to the k-th rule, $i = 1, 2, \cdots, 5$; $k_i = 1, 2, \cdots, 5$.

12.3.3 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" method is used in the proposed controller, combining the outputs represented by the implied fuzzy sets from all rules to generate 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

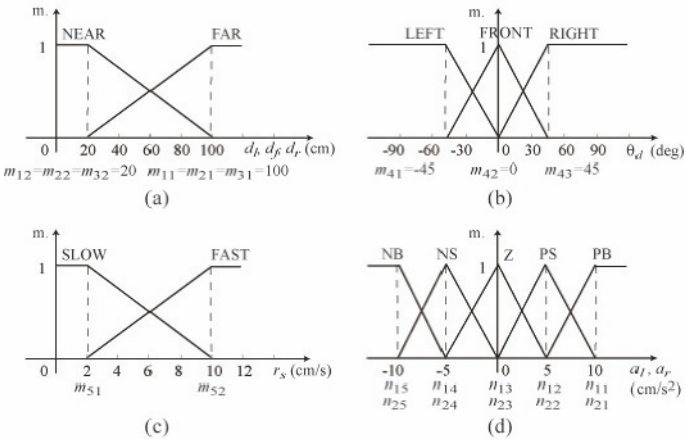

Fig. 12.4 Membership functions of the input and output variables, m_{ij}, n_{ls}: the centers of membership functions; m.: membership; meaning of NB to PB: see foot of (a) membership functions of obstacle distances; (b) membership functions of the target direction; (c) membership functions of the current robot speed; and (d) membership functions of the output variables accelerations of two wheels

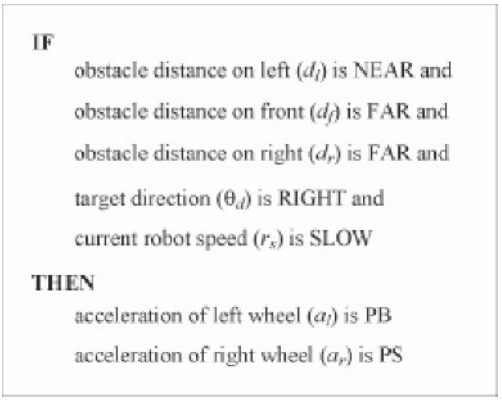

Fig. 12.5 An example of the inference rules. d_l = {NEAR, FAR}, d_f = {NEAR, FAR}, d_r = {NEAR, FAR}, θ_d = {LEFT, Center, RIGHT}, r_s = {SLOW, FAST}, a_l = {PB, PS, Z, NS, NB}, a_r = {PB, PS, Z, NS, NB}

$$a_l = \frac{\sum_{k=1}^{48} v_{k,1} q_k}{\sum_{k=1}^{48} q_k}, \tag{12.5}$$

$$a_r = \frac{\sum_{k=1}^{48} v_{k,2} q_k}{\sum_{k=1}^{48} q_k}, \tag{12.6}$$

where $v_{k,1}$ and $v_{k,2}$ denote the estimated values of the outputs provided by the k-th rule, which are related to the center of membership functions of the output variables.

12.3.4 The Physical Meanings of Variables and Parameters

The proposed model keeps the physical meanings of the variables and parameters during the processing, while the conventional fuzzy logic based model [11] cannot keep the physical meanings of variables.

The physical meaning of fuzzy variables is explained in Figure 12.3. In this chapter, 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 output, $s = 1, \cdots, 5$ represents number of the 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 used in (12.1), (12.2) and (12.3), $[y_1, y_2] = [a_l, a_r]$ is the output vector described in (12.5) and (12.6). The variable p_{ij} is the degree of membership for the i-th input corresponding to the j-th term of the input variable obtained by either (12.1), (12.2) or (12.3) according to different membership functions; q_k is conjunction degree of the IF part of the k-th rule obtained by (12.4); $w_{i,k}$ denotes the center of the membership function corresponding to the i-th input and the k-th rule, and 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 $w_{5,48} = m_{52}$; and $v_{k,l}$ are the estimated value of the outputs provided by the k-th 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 variable 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 $v_{48,2} = n_{25}$.

12.3.5 Algorithm to Tune the Model Parameters

To smooth the trajectory generated by the fuzzy logic model, a learning algorithm based on a neural networks 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 networks learning property. The vector of 21 parameters to be tuned in the proposed model is set as

$$\begin{aligned} Z = \{&m_{11}, m_{12}, m_{21}, m_{22}, m_{31}, m_{32}, m_{41}, m_{42}, m_{43}, m_{51}, m_{52}, \\ &n_{11}, n_{12}, n_{13}, n_{14}, n_{15}, n_{21}, n_{22}, n_{23}, n_{24}, n_{25}\}. \end{aligned} \tag{12.7}$$

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 shown in Figure 12.2. 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 chapter, the LMS algorithm is used to minimize the following criterion function.

$$E = \frac{1}{2} \sum_{l=1}^{2} (y_l - \hat{y}_l)^2, \tag{12.8}$$

where $[y_1, y_2] = [a_l, a_r]$ is the output vector described in (12.5) and (12.6); and $[\hat{y}_1, \hat{y}_2]$ is the desired output vector, obtained by a derivation of the desired trajectory that is manually marked on the simulator. Thus, the parameters would be adapted as

$$Z(t+1) = Z(t) - \varepsilon \frac{\partial E}{\partial Z}, \tag{12.9}$$

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

$$m_{ij}(t+1)=m_{ij}(t)-\varepsilon_m\frac{\partial E}{\partial m_{ij}},\begin{cases} i=1,\cdots,5,\\ j=1,2,3; \end{cases} \tag{12.10}$$

$$n_{ls}(t+1)=n_{ls}(t)-\varepsilon_n\frac{\partial E}{\partial n_{ls}},\begin{cases} l=1,2,\\ s=1,\cdots,5, \end{cases} \tag{12.11}$$

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. From (12.1)-(12.6), the corresponding expressions for the membership function centers of input and output variables are given as

$$\begin{aligned}\frac{\partial E}{\partial m_{ij}}&=\sum_{l=1}^{2}\left(\frac{\partial E}{\partial y_l}\frac{\partial y_l}{\partial q_k}\frac{\partial q_k}{\partial p_{ij}}\frac{\partial p_{ij}}{\partial m_{ij}}\right)\\ &=-2\sum_{l=1}^{2}\left[(y_l-\hat{y}_l)\frac{v_{k,l}\sum_{k=1}^{48}q_k-\sum_{k=1}^{48}v_{k,l}q_k}{(\sum_{k=1}^{48}q_k)^2}\right]\frac{\operatorname{sign}(u_i-m_{ij})}{\sigma_{ij}};\end{aligned} \tag{12.12}$$

$$\frac{\partial E}{\partial n_{ls}}=\sum_{l=1}^{2}\left(\frac{\partial E}{\partial y_l}\frac{\partial y_l}{\partial v_{k,l}}\frac{\partial v_{k,l}}{\partial n_{ls}}\right)=(y_l-\hat{y}_l)\frac{q_k}{\sum_{k=1}^{48}q_k}. \tag{12.13}$$

The iterative procedure for the adaptation of the parameters and for the minimization of the criterion function are summarized in Figure 12.6. 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 three steps). 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.

12.3.6 Algorithm to Suppress Redundant Rules

48 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.

After the training to tune the model parameters, the learning algorithm to suppress redundant rules is considered. It is clear from the Figure 12.3 that the variables $w_{i,k}$ and $v_{k,l}$, which can be obtained from the model parameters m_{ij} and n_{ls} described above in Section 12.3, determine the response of the fuzzy rules to the input signals. Every rule is related to a weight vector

$$W_k=\{w_{1,k},\cdots,w_{5,k},v_{k,1},v_{k,2}\},\quad k=1,2,\cdots,48. \tag{12.14}$$

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. 12.6 The algorithm to tune the parameters

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 Figure 12.7. 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 is obtained. Thus, the minimum number of nodes in the second layer of the structure in Figure 12.3 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.

12.3.7 State Memorizing Strategy

Usually, the fuzzy behavior-based robot, like other reactive robots, suffers from the "symmetric indecision" and the "dead cycle" problems. The proposed model resolves the "symmetric indecision" problem by designing several mandatory-turn rules (e.g. Rules 21, 22, 45, 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.

```
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. 12.7 The algorithm to suppress redundant rules

A typical "dead cycle" situation is shown in Figure 12.8(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 the 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 above "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 Figure 12.9. 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.

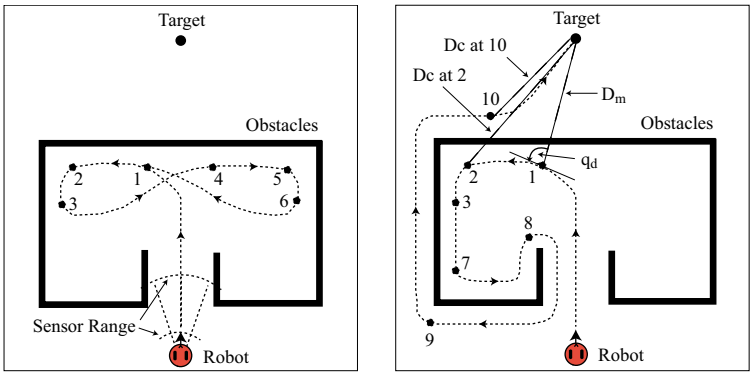

Fig. 12.8 Robot navigation in a "dead cycle" situation with limited sensor range. (a) without the proposed state memory strategy; and (b) with the proposed state memory strategy

The flow diagram of the algorithm is shown in Figure 12.10. 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 on the right side and the target direction θ_d shown in Figure 12.8(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 Figure 12.8(b). During State 1 or 2, the robot assumes the target just on the front left or right side of 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 the Figure 12.8(b). The robot changes back to State 0 when D_c at Position 10, shown in Figure 12.8(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.

12.4 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 Figure 12.1. 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^o. The sensor ring radius is 0.1 m. The radiation cone of the sensors is 25^o. The sensing range of the ultrasonic sensors is from 0.04 m 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 on-board robot sensors are limited.

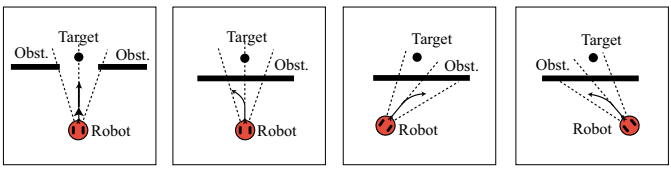

Fig. 12.9 (a) An illustration of robot states; (b) state 0; (c) state 1; and (d) state 2

12.4.1 Off-line Training Phase to Tune 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 situations for the mobile robot, the training environment should be designed in a relatively complicated way, 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 Figure 12.11(a).

According to the experience, the model parameters are initialized as

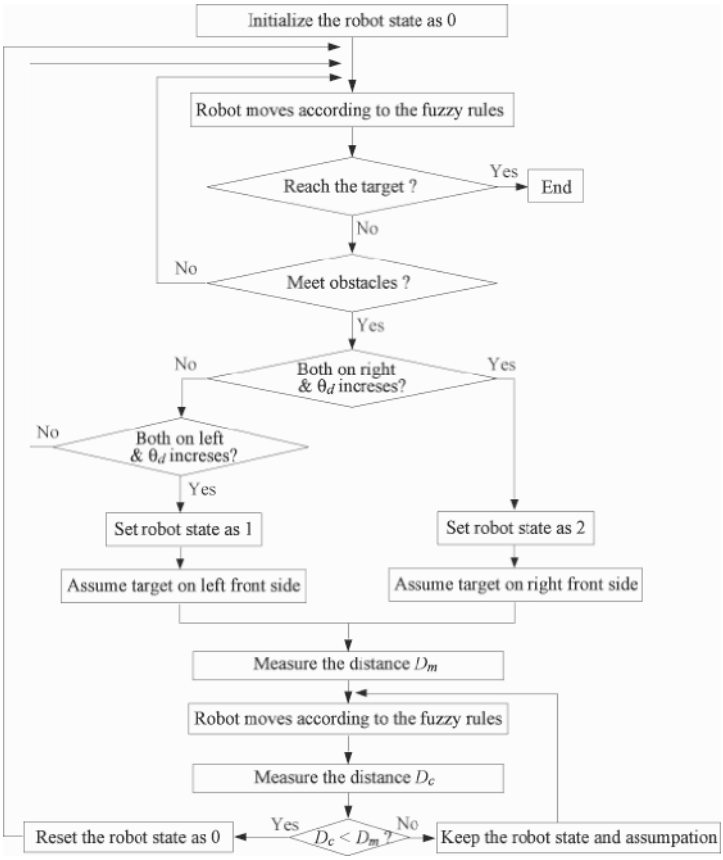

Fig. 12.10 Flow diagram of the developed state memorizing strategy

$$\{m_{11}, m_{12}, m_{21}, m_{22}, m_{31}, m_{32}, m_{41}, m_{42}, m_{43}, m_{51}, m_{52}\} = \{100, 20, 100, 20, 100, 20, -45, 0, 45, 2, 10\}; \quad (12.15)$$

$$\{n_{11}, n_{12}, n_{13}, n_{14}, n_{15}, n_{21}, n_{22}, n_{23}, n_{24}, n_{25}\} = \{10, 5, 0, -5, -10, 10, 5, 0, -5, -10\}; \quad (12.16)$$

$$\{\sigma_{11}, \sigma_{12}, \sigma_{21}, \sigma_{22}, \sigma_{31}, \sigma_{32}, \sigma_{41}, \sigma_{42}, \sigma_{43}, \sigma_{51}, \sigma_{52}\} = \{160, 160, 160, 160, 160, 160, 90, 90, 90, 16, 16\}. \quad (12.17)$$

To get the desired outputs $[\hat{y}_1, \hat{y}_2]$, a reasonable path is drawn by an expert first as shown in Figure 12.11(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. During the training phrase, the adaptation is done at every time interval. The trajectory during the training phase is shown in Figure 12.11(c), where the trajectory is winding at the beginning period, but smoother at the end period. As mentioned before, 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 as set in (12.7). In this simulation, after the learning, the better parameters of the membership functions are

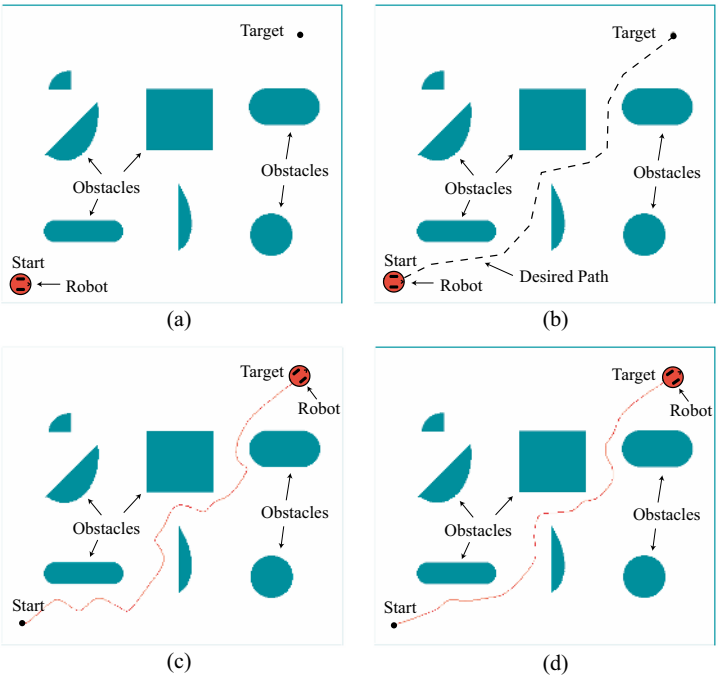

Fig. 12.11 The process of the training phrase. (a) the workspace; (b) the desired trajectory; (c) the trajectory using initial model parameters and during the training; (d) the trajectory using new model parameters got from the training

obtained as

$$\{m_{11}, m_{12}, m_{21}, m_{22}, m_{31}, m_{32}, m_{41}, m_{42}, m_{43}, m_{51}, m_{52}\} = \{94, 18, 105, 22, 95, 19, -48, 0, 47, 1.2, 9.4\}; \quad (12.18)$$

$$\{n_{11}, n_{12}, n_{13}, n_{14}, n_{15}, n_{21}, n_{22}, n_{23}, n_{24}, n_{25}\} = \{9.5, 5.4, 0, -5.5, -9.3, 9.4, 5.6, 0, -5.3, -9.4\}. \quad (12.19)$$

$$\{\sigma_{11}, \sigma_{12}, \sigma_{21}, \sigma_{22}, \sigma_{31}, \sigma_{32}, \sigma_{41}, \sigma_{42}, \sigma_{43}, \sigma_{51}, \sigma_{52}\} = \{160, 160, 160, 160, 160, 160, 90, 90, 90, 16, 16\}. \quad (12.20)$$

By using the new parameters, the mobile robot can work very well. The trajectory is shown in Figure 12.11(d). Figure 12.12(a) shows the smooth trajectories generated in other case by new model parameters in (12.18 and 12.19) after the training. But the trajectories are winding in Figure 12.12(b), if without the training phrase.

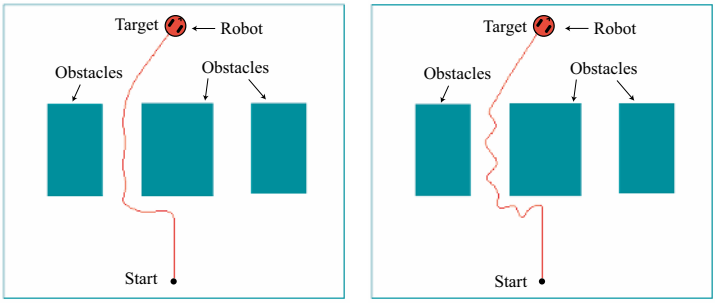

Fig. 12.12 Mobile robot trajectories (a) generating a smooth trajectory with the proposed learning algorithm; and (b) generating a winding trajectory without the learning algorithm

12.4.2 Off-line Training to Remove Redundant Rules

As mentioned above, every rule is related to a weight vector in (12.14). And the weight vectors depend on (12.18) and (12.19). Table 12.3 shows the weights related to the rules.

After the applying of the selection algorithm, the number of the rules will be less than 48, depending on the tolerance δ. Table 12.4 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 Figure 12.13 when δ = (1)0, (2)0.001, (3)0.005, (4)0.01. It is obvious that the trajectories in Figures 12.13(a), (b) and (c) are almost the same, except in Figures 12.13(d). This means that, with only 38 rules,

Table 12.3 The 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 12.4 The 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

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 12.2. According to the experience, the redundant rules are Rules 12, 14, 16, 18, 20, 22, 34, 40, 44 and 46 in Table 12.2. This can be explained from the rule based directly. For example, Rule 40 is similar to Rule 39, and is therefore redundant.

12.4.3 Effectiveness of the State Memory Strategy

In this section, after suppressing the redundant rules, the proposed controller is applied to the typical complicated situation that causes the "dead cycle" problem existing in many conventional approaches.

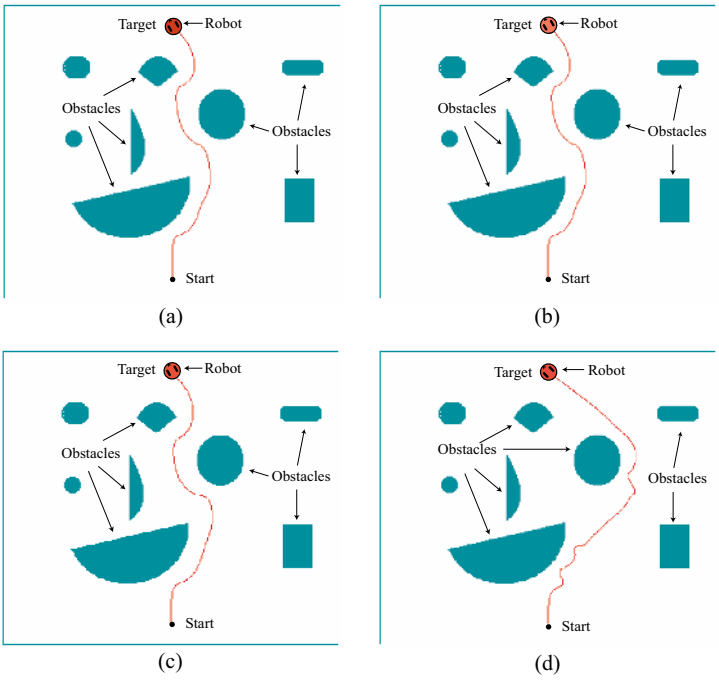

Fig. 12.13 Robot trajectories with different number of rules when the tolerance δ is selected (a) 0 with 48 rules; (b) 0.001 with 47 rules; (c) 0.005 with 38 rules; and (d) 0.01 with 26 rules

A robot moving in a U-shaped obstacles situation is shown in Figure 12.14. 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 reaches the target. It shows that the robot trajectory from the start position to the target does not suffer from the "dead cycle" problem.

12.4.4 Dynamic Environments and Velocity Analysis

Figure 12.15(a) shows the trajectory of the robot in a dynamic environment that the target moves in a cycle. Assume a target is moving in a cycle from the position (14, 6) around the point (10, 6) with 4 radius and clockwise direction, while the robot starts from position (10, 18) in the workspace and is moving to the right. At the beginning, the robot turns left, and then follows the target with a smooth trajectory.

The recorded velocity profile of both wheels in this simulation is presented in Figure 12.15(b). It can be seen from this figure that at the beginning 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 during its navigation procedure.

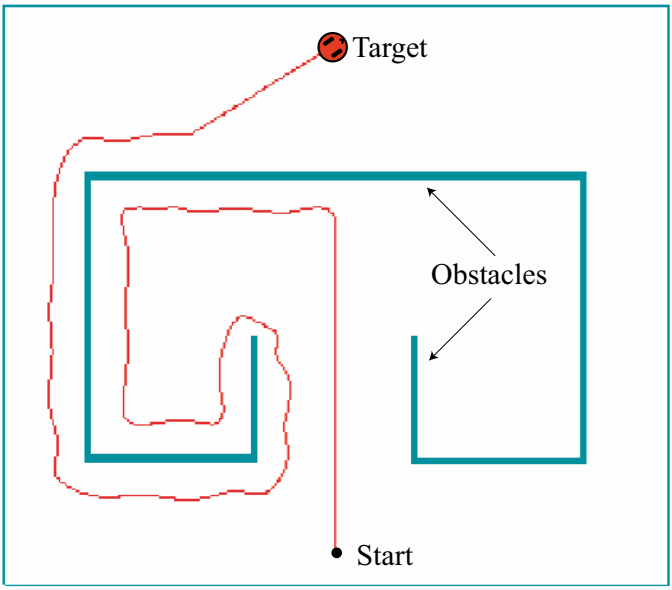

Fig. 12.14 Robot navigation in a complicated situation without suffering from the "dead cycle" problem

In the second case, the robot navigation in a complicated environment is demonstrated in Figure 12.16(a), where the robot meets some static or movable obstacles and the target is also movable. Assume that the target moves in a line from the position (5, 5) to (16, 5) and then back to (5, 5); besides some static obstacles, one robot considering an obstacle is moving in a line from the position (3, 12) to (16, 12) and back to (3, 12); one robot is moving in a cycle from the position (15, 9) around the point (11.7, 9) with a 3.3 radius and an anticlockwise direction; another robot is moving randomly; and the controlled robot with a direction to the right starts from position (10, 18) in the workspace. A smooth trajectory is executed in obstacle avoidance and travelling to the target.

The recorded velocity profile of both wheels in this simulation is presented in Figure 12.16(b). It can be seen from this figure that at the beginning, the velocities of both wheels of the controlled robot increases; and the velocity of the right wheel increases and the velocity of the left wheel decreases when the robot turns to the left to avoid obstacles. The velocity increases or decreases a little when the robot makes a small turn, and changes much more for large turns.

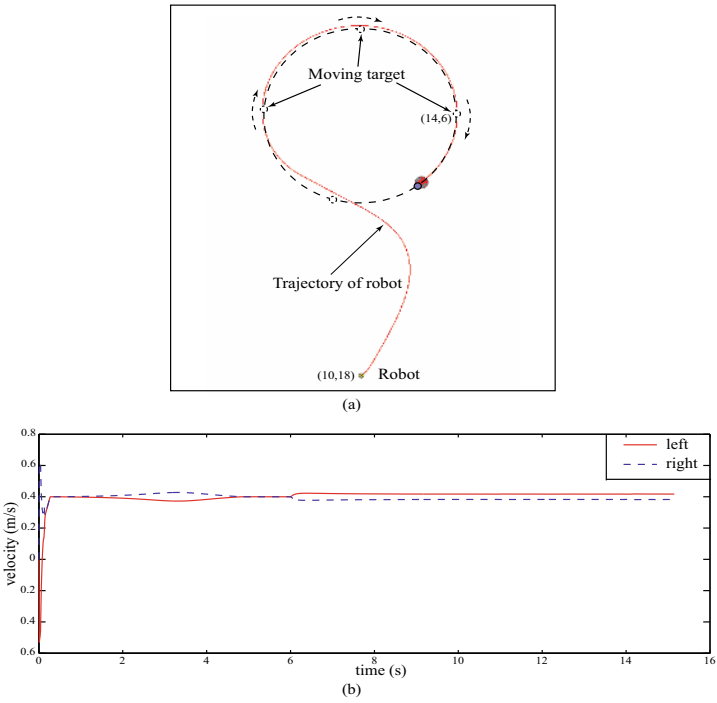

Fig. 12.15 Robot navigation in a dynamic environment with a moving target along a cycle. (a) the generated trajectory; (b) the velocities of the robot

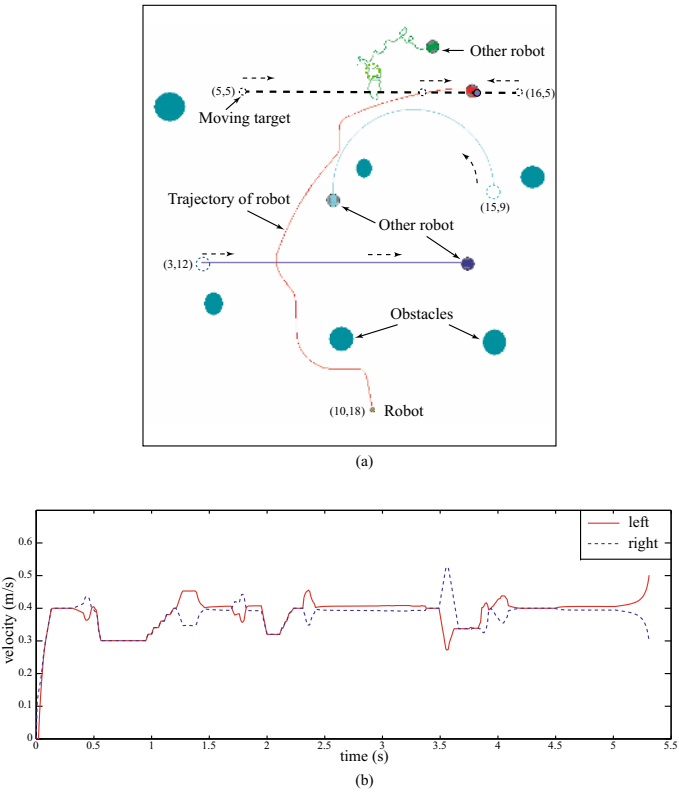

Fig. 12.16 Robot navigation in a dynamic environment with a moving target and moving obstacles. (a) the generated trajectory; (b) the velocities of the robot

12.5 Experiment of Studies

As a test bed, real robots were employed to test the performance of the proposed neuro-fuzzy approach, which navigated autonomously in an unknown environment using onboard sensors.

The shape of robots used as a test bed is shown in Figure 12.17. The left one includes the respective mechanical structure, electronic modules as well as the software development kit (SDK). The mechanical structure is already pre-built, including two 70mm diameter separated driving wheels with the maximum speed 9 meter per minute, animated head system, five servos, two direct current (DC) motors and so on. The electronic system is setup with a multimedia controller (PMB5010), a sensing-motion controller (PMS5005) and various peripheral electronic modules, includes one color image module with camera, four ultrasonic range sensor modules, two pyroelectric human motion sensor modules, one tilt/acceleration sensor model, one ambient temperature sensor module, one sharp infrared distance mea-

suring sensor module, four rotary sensor modules, one wireless communication module, one microphone, one speaker, one battery pack, and so on. The software component is installed on a PC, being responsible to establish a wireless connection and exchange data with the robot. The users can develop their own applications in VC++ or VB using application programming interface (API), which can access the sensor information, send control commands, and configure the system settings. The right robot is the upgraded model of the left one.

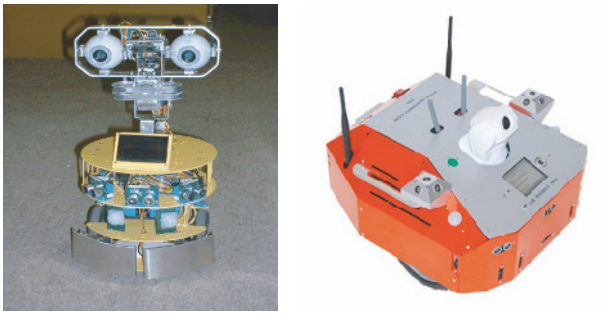

Fig. 12.17 Real robots used as a test bed

Two issues are worth mentioning here. First, because of the limitation of the equipments of the robot, only three ultrasonic sensors with the range between 0.04 m to 2.55 m are used to get the obstacle distances from the robot at left, in front, and at right. The direction of the target, and the distance between the robot and the target, are ignored. In this situation, the robot can wander on a level floor with obstacle avoidance using the proposed neuro-fuzzy approach. Secondly, because of the sensor noise and the misreading of the sensors, a simple sensor fusion algorithm is used to filter the noise of sensors. Based on these situations, two experiments in a variety of environments are given as follows.

A situation of the robot avoiding the obstacle when it is encountered on the left side of the robot is shown in Figure 12.18. Figure 12.18(a) shows the simulation trajectory of the robot in this situation, where the robot turns right to avoid the obstacle on the left side, then goes straight and turns left when it meets obstacles on the right side. Figures 12.18(b), (c), (d), and (e) show the pictures of the real robot at position 1, 2, 3, and 4 during its navigation. Figure 12.18(f) presents the recorded sensor profiles of left, front and right sensors of the robot in this experiment. Figure 12.18(g) presents the recorded velocity profiles of left and right wheels of the robot in this experiment.

A situation in which the robot avoids the obstacle when the obstacle is encountered in front of the robot is shown in Figure 12.19. Figure 12.19(a) shows the simulation trajectory of the robot when it turns left to avoid the obstacle in front of robot, then goes straight and turns left again when it meets obstacles on the right side. Figures 12.19(b), (c), (d), and (e) show the pictures of the real robot at position

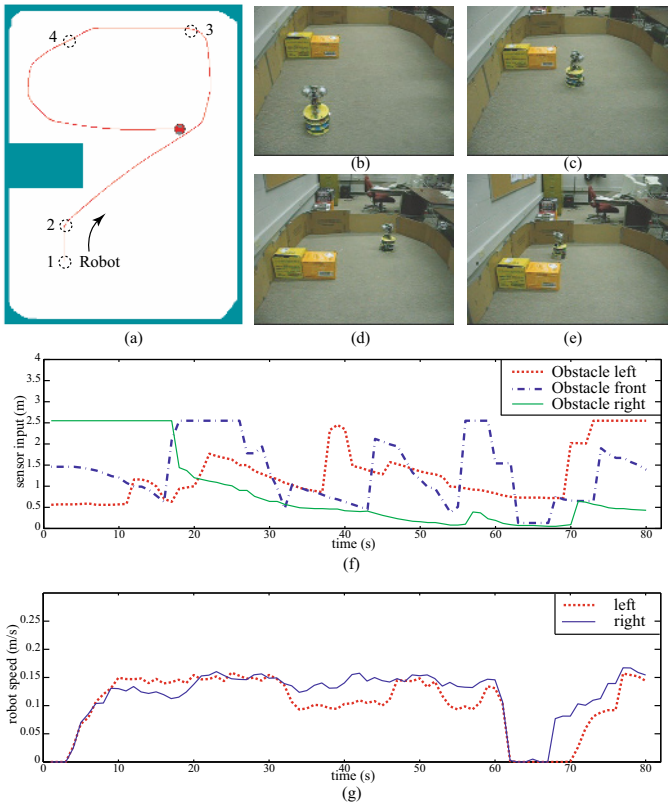

Fig. 12.18 Robot moves when obstacles on its left (a) trajectory; (b)-(e) snapshots; (f) sensor input; and (g) robot speed

1, 2, 3, and 4 during its navigation. Figure 12.19(f) presents the recorded sensor profiles of left, front and right sensors of the robot in this experiment. Figure 12.19(g) presents the recorded velocity profiles of left and right wheels of the robot in this experiment.

From the movies of the experiments, we can see that the robot can go smoothly without obvious oscillation in a workspace with different situations. From the robot speed analysis of the experiments, we can see that the speeds of the left and right wheels of the robot are smooth. In general, "smoothness" is synonymous to "having small high-order derivatives". Smoothness is relative. It can be seen obviously from a picture, or defined as the change less than a value. In this study, the "smoothness" is defined as without obvious oscillation. It can be defined as that the speed change is less than 0.05 m/s^2 and the angle change is less than 30 $degree/s$.

Comparing the performance between the experiment results and and the simulation results, the velocity of the simulation is smoother than the experiment. That is reasonable, because the input information, and the velocity control of the simulation

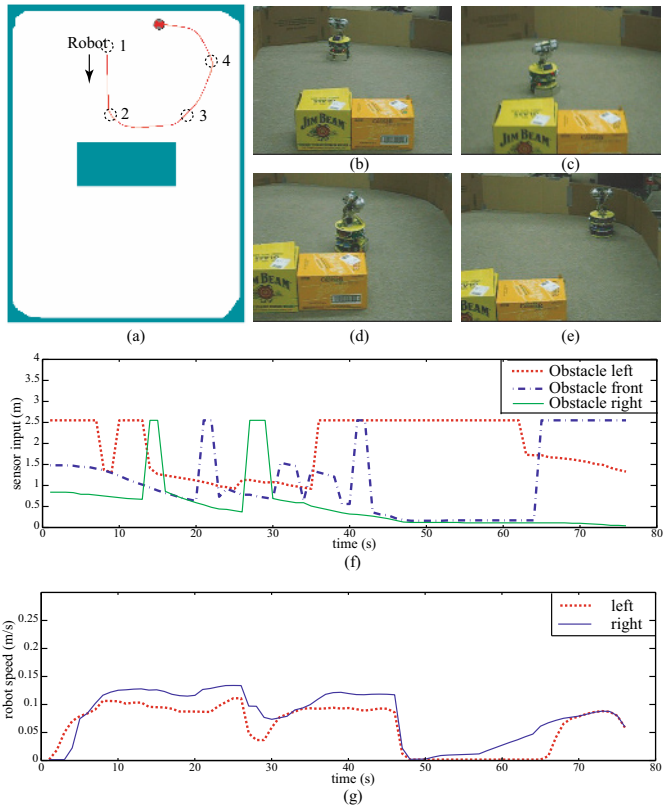

Fig. 12.19 Robot moves when obstacles are in front of it (a) trajectory; (b)-(e) snapshots; (f) sensor input; and (g) robot speed

are perfect, but in a practical robot, the sensor noise, and the physical moving of the real robot will infect the performance.

12.6 Summary

In this chapter, a novel fuzzy logic based control system, combining sensing and a state memory strategy, was 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.

- The proposed model keeps the physical meanings of the variables and parameters during the processing, while some conventional models cannot.

- 48 fuzzy rules and three behaviors are designed in the proposed model, much fewer than conventional approaches that use hundreds of rules.
- 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.
- The proposed selection algorithm can automatically suppress redundant fuzzy rules when there is a need.
- A very simple yet effective state memory strategy is designed. It can resolve the "dead cycle" problem existing in some previous approaches without changing the fuzzy control rule base.

References

1. Aguirre E, Gonzalez A (2003) A fuzzy perceptual model for ultrasound sensors applied to intelligent navigation of mobile robots. Applied Intelligence, 19:171-187.
2. Chatterjee A, Matsuno F (2007) A neuro-fuzzy assisted extended kalman filter-based approach for simultaneous localization and mapping (slam) problems. IEEE Transactions on Fuzzy Systems, 15(5):984-997.
3. Filliat D, Meyer JA (2003) Map-based navigation in mobile robots: a review of localization strategies I. Cognitive Systems Research, 4:243-282.
4. Gao J, Xu D, Zhao N, Yan W (2008) A potential field method for bottom navigation of autonomous underwater vehicles. Intelligent Control and Automation, 7th World Congress on WCICA, 7466-7470.
5. Hagras H, Callaghan V, Colley M (2000) Online learning of the sensors fuzzy membership functions in autonomous mobile robots. Proceedings of the 2000 IEEE International Conference on Robotics and Automation, 3233-3238, San Francisco.
6. Hong J, Choi Y, Park K (2007) Mobile robot navigation using modified flexible vector field approach with laser range finder and ir sensor. International Conference on Control, Automation and Systems, 721-726.
7. Kim CT, Lee JJ. (2005) Mobile robot navigation using multi-resolution electrostatic potential field. 31st Annual Conference of IEEEIndustrial Electronics Society, 1774-1778.
8. Kim MY, Cho H (2004) Three-dimensional map building for mobile robot navigation environments using a self-organizing neural network. Journal of Robotic Systems, 21(6):323-343.
9. Li H, Yang SX (2003) A behavior-based mobile robot with a visual landmark recognition system. IEEE Transactions on Mechatronics, 8(3):390-400.
10. Machado C, Gomes S, Bortoli A, *et al.* (2007) Adaptive neuro-fuzzy friction compensation mechanism to robotic actuators. Seventh International Conference on Intelligent Systems Design and Applications, 581-586.
11. Marichal GN, Acosta L, Moreno L, *et al.* (2001) Obstacle avoidance for a mobile robot: A neuro-fuzzy approach. Fuzzy Sets and Systems, 124(2):171-179.
12. Na YK, Oh SY (2003) Hybrid control for autonomous mobile robot navigation using neural network based behavior modules and environment classification. Autonomous Robots, 15:193-206.
13. Noborio H, Nogami R, Hirao S (2004) A new sensor-based path-planning algorithm whose path length is shorter on the average. IEEE International Conference on Robotics and Automation, 2832-2839, New Orleans, LA.
14. Nogami R, Hirao S, Noborio H (2003) On the average path lengths of typical sensor-based path-planning algorithms by uncertain random mazes. IEEE International Symposium on Computational Intelligence in Robotics and Automation, 471-478, Kobe, Japan.

15. Park K, Zhang N (2007) Behavior-based autonomous robot navigation on challenging terrain: A dual fuzzy logic approach. IEEE Symposium on Foundations of Computational Intelligence, 239-244.
16. Rusu P, Petriu EM, Whalen TE, et al (2003) Behavior-based neuro-fuzzy controller for mobile robot navigation. IEEE Transactions on Instrumentation and Measurement, 52(4):1335-1340.
17. Song KT, Sheen LH (2000) Heuristic fuzzy-neuro network and its application to reactive navigation of a mobile robot. Fuzzy Sets and Systems, 110(3):331-340.
18. Sun F, Li L, Li HX, et al (2007) Neuro-fuzzy dynamic-inversion-based adaptive control for robotic manipulators- discrete time case. IEEE Transactions on Industrial Electronics, 54(3):1342-1351.
19. Ulrich I, Borenstein J (2000) Vfh*: Local obstacle avoidance with look-ahead verification. Proceedings of the 2000 IEEE International Conference on Robotics and Automation, 2505-2511, San Francisco, CA.
20. HaiPeng Xie (2008) Wireless networked autonomous mobile robot-i90 sentinel2 user guide. Available at http://www.drrobot.com/products/item_downloads/Sentinel2_1.pdf.
21. Yang SX, Meng QH (2003) Real-time collision-free motion planning of mobile robots using neural dynamics based approaches. IEEE Transactions on Neural Networks, 14(6):1541-1552.
22. Yang SX, Moallem M, Patel RV (2003) A novel intelligent technique for mobile robot navigation. IEEE Conference on Control Applications, 674-679.
23. Zhu A, Yang SX (2004) A fuzzy logic approach to reactive navigation of behavior-based mobile robots. IEEE International Conference on Robotics and Automation, 5:5045-5050.
24. Zhu A, Yang SX (2007) Neurofuzzy-based approach to mobile robot navigation in unknown environments. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 37(4):610–621.