A Neuro-Fuzzy Controller for Mobile Robot Navigation and Multirobot Convoying

Kim C. Ng, *Member, IEEE,* and Mohan M. Trivedi, *Senior Member, IEEE*

***Abstract*—A Neural integrated Fuzzy conTroller (NiF-T) which integrates the fuzzy logic representation of human knowledge with the learning capability of neural networks is developed for nonlinear dynamic control problems. NiF-T architecture comprises of three distinct parts: 1) Fuzzy logic Membership Functions (FMF), 2) a Rule Neural Network (RNN), and 3) an Output-Refinement Neural Network (ORNN). FMF are utilized to fuzzify sensory inputs. RNN interpolates the fuzzy rule set; after defuzzification, the output is used to train ORNN. The weights of the ORNN can be adjusted on-line to fine-tune the controller. In this paper, real-time implementations of autonomous mobile robot navigation and multirobot convoying behavior utilizing the NiF-T are presented. Only five rules were used to train the wall following behavior, while nine were used for the hall centering. Also, a robot convoying behavior was realized with only nine rules. For all of the described behaviors—wall following, hall centering, and convoying, their RNN's are trained only for a few hundred iterations and so are their ORNN's trained for only less than one hundred iterations to learn their parent rule sets.**

***Index Terms*—Convoying behavior, fuzzy logic controller, hall centering behavior, learning, mobile robot navigation, neural networks, sensor-driven robots, team robotics, wall following behavior.**

I. Introduction

In designing autonomous robotic systems, two important challenges are frequently encountered. The first deals with the nonlinear, real-time response requirements underlying the sensor–motor control formulation. The second deals with how to model and use the approach that a human will take for such a problem. Often the human experience and approach can best represented with a set of linguistic rules. Fuzzy logic controllers can mimic experts. Nevertheless, deriving and fine-tuning the entire rule set and membership functions is often tedious and difficult. Neural controllers learn, yet discrete input representations may cause such systems to be unstable. In addition, sufficient training patterns are usually difficult to obtain, and training time for the whole dynamic range is very long. We develop a control architecture blending neural networks and fuzzy logic is developed for nonlinear control problems. It takes advantage of the best of fuzzy logic and neural networks—assimilating human expertise with continuous representation, combining with learning capability.

Manuscript received March 17, 1996; revised February 24, 1998.

The authors are with the Department of Electrical and Computer Engineering, University of California, San Diego, CA 92093-0407 USA (e-mail: kimng@swiftlet.ucsd.edu).

Publisher Item Identifier S 1083-4419(98)07296-3.

Robust and practical control motivates us to develop a general architecture for integrated sensor-based robotic systems which has the following capabilities.

- Integrated sensing, control, and actuator modules.
- Real-time performance.
- Ability to successfully handle noisy sensor signals.
- Reactive controller design which captures high-level, linguistically based human expertise in a set of fuzzy rules.
- Training of neural networks directly with fuzzy rules instead of numerical sample data. The linguistic expert rules are easier to obtain and more reliable.
- Learning capability to interpolate new sets of rules for robust and smooth operation of the system. New rules are derived of parent rules as specified by a human so that a minimum of rules are extracted from the expert.
- Learning capability to refine the performance of the integrated systems.
- General applicability to a wide range of sensor-driven robotic systems.

We use the navigation and convoying as representatives of a class of nonlinear control problems involving noisy sensory signals and real-time feedback requirements. Analytical models for mobile robot navigation and especially multiple mobile robots moving in formation are difficult to obtain. The Neural integrated Fuzzy conTroller (NiF-T) is successfully applied to solve these nonlinear problems.

II. Architecture of NiF-T

A. Related Approaches

Fuzzy logic can be utilized for the fusion of multiple neural networks. In designing fuzzy logic expert systems, a great deal of care and effort is required to obtain the rules. This problem can be attacked by developing basic fuzzy neurons to get the membership functions and rules. Pedrycz has introduced two aggregation neurons named AND and OR, based on the logic-oriented processing mechanisms of fuzzy sets in [1]. These neurons can be aggregated into a single computational structure to generate a collection of if-then statements describing input/output relationships. With the use of neural methods, membership functions and rules may eventually be automatically generated and tuned. The fuzzy-neural networks in [2] identified the fuzzy model of a nonlinear system automatically. FLIP-net can detect rule deficiencies and

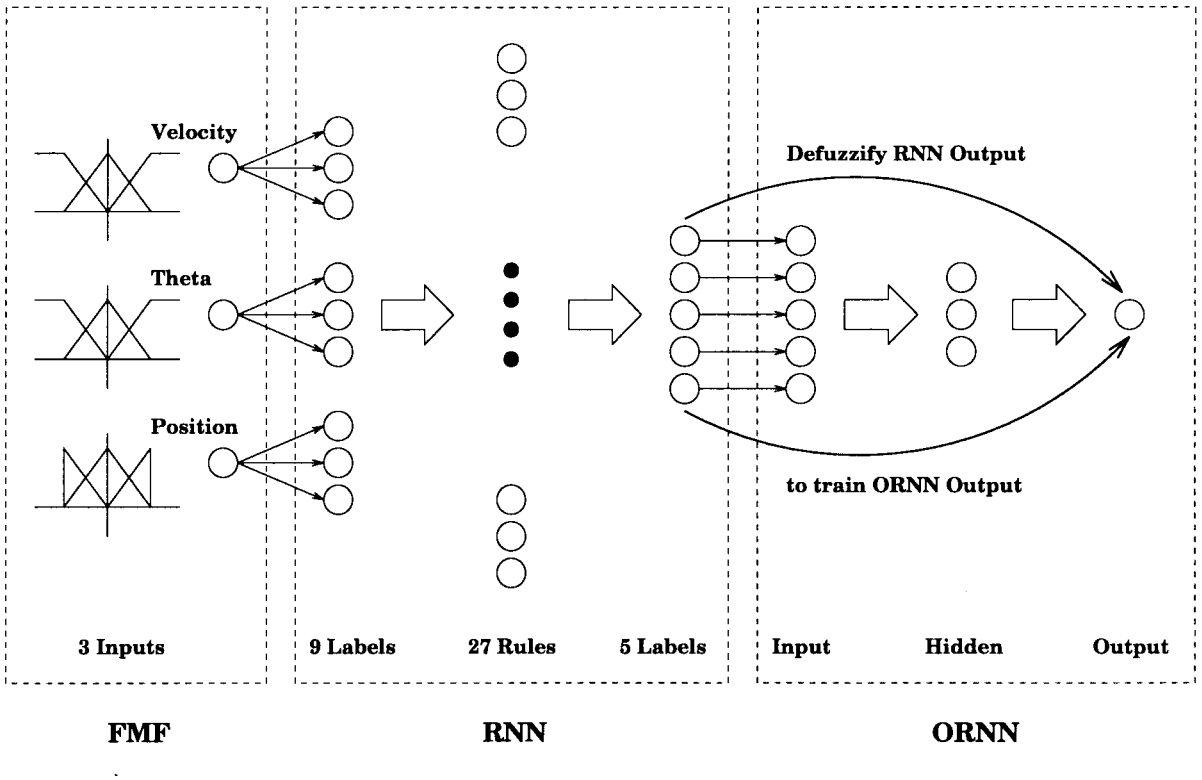


Fig. 1. The NiF-T model and its three main modeules: Fuzzy logic Membership (FMF), Rule Neural Network (RNN), and Output-Refinement Neural Network (ORNN).

tune membership functions [3]. Using neural networks to solve the problems with large-scale fuzzy knowledge bases may, however, "change the logical implications of AND and OR logic during the learning process [3]." This diminishes two major advantages of fuzzy logic, readability and maintainability. Identifying and tuning the fuzzy inference rules and membership functions simultaneously is an unsolved problem.

When constructing a controller, two kinds of information are available: numerical data from sensors and linguistic information from human experts. NiF-T does not utilize numerical data in the same way such as Higgins's [4] or Ishibuchi's methods [5] to generate rules, because converting an existing controller into a fuzzy controller is not desired. Instead, we strive to maintain the original fuzzy logic formulation so that its major advantages can be preserved.

For the membership functions, a method similar to Ishibuchi's is used. Fuzzy numbers are propagated through neural networks, unlike Horikawa [2], who constructs membership functions using neurons. Bouslama [6] has used a three-layer neural network for the fuzzy control problem. However, instead of training the controller directly with rules, his training vectors were discrete data from a reference fuzzy logic controller. Our architecture is also different from fuzzy neurons and fuzzy min-max neural networks. A small number of rules are used by NiF-T. In our approach, tuning membership functions and rules are not of interest. Therefore, there is no physical output membership layer connected with variable weights for defuzzification such as in Higgins and Berenji [7] have. The hidden and output layers of our RNN perform the fuzzy logical operations, (such as min-max), and interpolate the rest of the untrained rules. Basically, after RNN has been trained, it acts like a reference controller. A smaller network, ORNN, copies the behavior of RNN and adjusts the controller performance using an optimization criterion.

B. Architecture

The NiF-T model shown in Fig. 1 has three main parts: Fuzzy logic Membership Functions (FMF), Rule Neural Network (RNN), and Output-Refinement Neural Network (ORNN). Piecewise trapezoidal functions or any other functions which can map $\Re \rightarrow [0, 1]$ can be used for FMF. The number of sets of membership functions is the same as the number of sensory input variables, and the number of labels of each set of membership functions determines the number of RNN input nodes. For instance, in Fig. 1, three input variables, each having three labels (three membership functions), require nine RNN input nodes. The hidden layer of the RNN can have any number of nodes. The number of RNN output nodes is fixed by the number of output labels. The ORNN's inputs are the RNN's outputs. The ORNN output layer always has a single neuron.

C. Training and Testing Procedures

Both the RNN and ORNN are multilayer feedforward neural networks using *Back Propagation* (BP). RNN maps input vectors to output vectors. RNN's objective function is defined

by its actual outputs and the corresponding target outputs (read Section II-D for more details). After RNN is trained satisfactorily with the FMF and ORNN decoupled, training ORNN follows. There are two training phases for ORNN: the first is off-line, and the second is on-line. During the first phase, the FMF is temporarily decoupled, but the RNN is connected to the ORNN with the RNN's weights frozen as shown in Fig. 1. The same input training vectors for the training of RNN are again fed into the RNN; however, the ORNN's target training output is the defuzzified output of the RNN output vector (the defuzzification process is discussed in Section II-E). For the second phase, FMF, RNN, and ORNN are all connected as in Fig. 1. ORNN's on-line training objective function is to minimize controller errors based on the continuous feedback sensory inputs, using the BP with objective functions defined in (1). For both phases, the weights of RNN are frozen. Only the ORNN can be selected to be trained further.

The following is a brief summary of the overall procedure:

Step 1. Determine the input parameters: the number of sensory inputs (speed, distance, $\cdots$), and the typical parameters for a BP neural network.

Step 2. Construct membership functions and define fuzzy rules.

Step 3. Conduct preliminary training.

a) Train the RNN with the FMF and ORNN decoupled. The fuzzy rules (input/output pairs) are specified as discussed in Section II-D.

b) Train the ORNN with the RNN connected, but without the FMF. The same input pattern in (a) are passed through the RNN; however, the ORNN desired output patterns are the RNN defuzzified crisp output. Train the ORNN to output defuzzified instances of RNN outputs. Since both training in (a) and (b) are with the FMF decoupled, the shifting and shape change in the input membership functions will **not** require retraining the RNN and ORNN. The shifting has to preserve the same order of the RNN input neurons.

Step 4. Test the controller with integrated sensory inputs.

a) Without on-line training: pass the sensory inputs through the integrated FMF, RNN, and ORNN. The RNN and ORNN were trained in the previous steps. The output from the ORNN is to control a motor.

b) With on-line training: pass the inputs as in (a). Change ORNN's weights according to the objective function based on the feedback sensory inputs. Note that the target output is not from the RNN defuzzified output anymore. The objective function is defined as follows:

$$J = \frac{\sum_{i=1}^{N} c_i \epsilon_i^2}{N} \tag{1}$$

where c_i are constant coefficients. ϵ_i are the difference between desired and current state of the feedback sensory inputs. The N is the number of sensory inputs. The greater influence of an input variable on the system, the larger relative value of its associate coefficient c_i is. For output refinement purpose, set all c_i less than one. Otherwise, the output error will be amplified, and the weights of the network will be modified in large step. Cheng [8] has also used the similar training technique for his AGV steering.

Step 5. If the number of membership functions in FMF (the number of RNN input nodes changes accordingly) or the number of training rules is varied, or the consequence order of a rule is altered, repeat from Step 3. Otherwise, if only the weighting factor of the RNN's output label is changed, repeat from Step 3(b). See Section II-E for the discussion of the weighting factor.

If the trained weights from Step 4(b) are to be loaded onto Step 4(a), we must be sure that they are successfully trained at the instance the controller quits from Step 4(b); then the controller will have an improved version for that particular environment. Additionally, we should not train the controller in a long irregular environment using Step 4(b). To eliminate unintentional disturbances (errors that are not due to the controller itself, such as a sudden environmental change or discontinuity), a counter can be set to record the number of times the controller approaches its control goal. Increase the counter when the goal is reached; decrease the counter otherwise. The counter is initialized with a positive number that depends on the confidence that we have in our developed rule set at the beginning of Step 4(b). The controller starts to learn again if the counter drops to zero. This allows the controller time to verify its current weights. In addition, this enables the ORNN to serve as an output refinement network and to prevent the network from overfitting [9] (keep the controller from overreacting to every action).

D. Design of the Rule Neural Network (RNN)

We consider two types of rules in a rule set: parent and derived [4]. The RNN is used to learn the parent rules and then interpolate or generate the derived rules.

Knowledge of the system is the key to generating rules. In developing the entire rule set, we must always keep the control objective in mind. A whole set of rules could be developed. If a minimal number of rules are to be created manually, we must investigate which rules are the parent rules. Boundary conditions in differential equations suggest that the parent rules are boundary rules. A stable system is a controller's goal, so the rule at which all inputs and the output are zero must be one of the boundary rules. How a human balances a ball or drives a car gives us a key to understanding this idea. If a person, for example, can balance a fast-rolling ball at one end of the beam on a highly inclined theta, he/she should be able to handle intermediate situations as well. Therefore, the boundary rules governs situations when the system is most unstable and critical.

Table I shows an example of the boundary rules. Rule 1 is the stabilized rest position. Rules 2–5 are boundary rules. Its corresponding patterns for the training of RNN are shown in Table II. The numbers 1 and 0 are the membership values. For Rule 1, the membership value of the label z (m_z) for both the inputs of *Road Offset* and *Orientation Error* are one, so is Rule 1 target output $m_z = 1$. RNN is trained with these input/output

TABLE I
THE WALL FOLLOWING PARENT RULE SET

Rule	Inputs		Output	Cond.
	Road Offset	Orient. Error		
1	z	z	z	Stable
2	p	p	-l	Critical
3	p	n	z	
4	n	p	z	
5	n	n	l	

TABLE II
RNN IS TRAINED WITH THESE PATTERNS

Rule	Inputs						Target Outputs		
	Road Offset			Orient. Error					
	n	z	p	n	z	p	-l	z	l
1	0	1	0	0	1	0	0	1	0
2	0	0	1	0	0	1	1	0	0
3	0	0	1	1	0	0	0	1	0
4	1	0	0	0	0	1	0	1	0
5	1	0	0	1	0	0	0	0	1

membership value pairs. For Rule 1, the inputs to the RNN six input nodes are (0 1 0 0 1 0); and the desired outputs are (0 1 0) at the RNN's three output nodes. The second training pair from Rule 2 is then (0 0 1 0 0 1) and (1 0 0). Both inputs and outputs are fuzzy membership vectors. When input membership values fall between 0 and 1, RNN will interpolate the outputs after it is trained.

E. Design of the Output-Refinement Neural Network (ORNN)

There are three main reasons for having the ORNN. First, the knowledge that is embedded in the RNN will not be altered when learning continues. Second, a smaller network is faster. Third, a single neuron at the output layer eases the credit assignment problem.

The ORNN is trained for two phases, off-line and on-line. During off-line training, the ORNN's inputs are connected directly from the outputs of the RNN with the FMF decoupled. The same input training patterns for RNN are again passed through the RNN with the RNN's trained weights frozen. The defuzzified output of the RNN is now the ORNN's target output. The RNN's output membership function label sequence is in accordance with the discussion in Section II-D. For defuzzification, a singleton method [4] described in (2) is used. The V_i is the output membership values at the RNN output node i; and the w_i is the fixed assigned weighting factor of that output membership function. However, the method described here is different from the Higgins and NeuFuz4 method [4], [10]. NiF-T does not have a physical layer for output membership functions in the RNN, and the weights w_i are all assigned and fixed, based on the robot physical operating limits (in this paper is either our robot's motor steering angle or speed; for instance, the robot, SMAR-T, can move at the maximum speed of 0.4 ft/s). The crisp output of RNN is calculated as follows:

$$O = \frac{\sum w_i V_i}{\sum V_i}. \tag{2}$$

Use the same example from Table II. If the RNN three output nodes [in the sequence of (−l z l)] have the values of (0.5, 0, 1), they can be written in mathematics as $V_1 = m_{-l} = 0.5$, $V_2 = m_z = 0.3$, and $V_3 = m_l = 1$. With the assigned weighting factors of $w_1 = -60$, $w_2 = 0$, and $w_3 = 60$, the defuzzified output of the RNN is 16.67.

III. MOBILE ROBOT NAVIGATION USING NiF-T

It is not uncommon to encounter studies dealing with a simple navigation problem, like keeping the vehicle in the lane center, would require thousands of iterations to train the wide dynamic range of thousands of sample data [11]–[13]. However, with the methods described in the previous section, the robot can now follow the wall and center itself in the hallway with only a few rules. Although the navigation environment that we choose is indoor rather than outdoor, the controller developed here can be applied to the Intelligent Vehicle Highway System (IVHS) with just a matter of changing the input sensory information, such as using cameras to detect the road boundary. The hall centering needs two side-wall information, while the wall following needs just one. Therefore, when applying to the case of IVHS road navigation, if one side of the road boundary information is missing while the robot is centering itself on the road, the robot should still be able to navigate safely with the backup of road following behavior. Of course, the robot performing the wall following or hall centering behavior has its own practical uses itself, such as vacuuming the floor or mowing the yard. We emphasize here that the NiF-T can be utilized to control a variety of nonlinear behaviors, both for indoor or outdoor applications.

A. Problem Statement

Two behaviors, wall following and hall centering, are realized in this section. The robot is supposed to hug wall at any specified distance and to center itself in a hallway for the wall following and hall centering behaviors, respectively.

There are three basic questions for the problem of navigation: "where is the robot?," "where is it going?," and "how should it get there?" [14]. A rotary sonar sensor is used to acquire the localization of a robot. After processing the sensor information, the proposed NiF-T in previous sections tells the robot where and how to go. The speed of the robot is set constant here. Only the orientation of the robot is changing continuously to reach the control goal.

B. Related Approaches

In order for a robot to navigate autonomously without running into any obstacle and to reach its designated destination, there are three major problems to consider. The robot must recognize landmarks in order to know its localization, and it must recognize obstacles in order to navigate safely. A related but somewhat more complicated behavior is that of obstacle negotiating. In this not only the obstacle needs to be detected but the robot should climb over it rather than avoid it. Chen and Trivedi [15] described a controller for a sonar-based tracked mobile robot for obstacle negotiating. A goal-driven robot needs path-planning, and then a controller tells the robot

how to get there. In the literatures, [16], [17], the components required for the intelligent robotics systems were thoroughly discussed. Generally four functional modules comprise an intelligent system: 1) sensing; 2) perception/planning and control; 3) motor; and 4) workspace. In this section, we focus mainly on designing a controller for the intelligent robotic system using a sonar sensor with the motor actuators.

Researchers have attempted to use fuzzy logic and neural networks to solve these problems. MORIA [18] had two rule blocks. One was used to recognize the perceptual environment, the other to drive the robot. Lee's AGV [19] avoided static and moving obstacles and navigates from a starting point toward the target using six fuzzy logic control modules. Although Fukuda has applied fuzzy template matching [20] along with a neural network to detect the ceiling landmark, fuzzy logic basically has been acting more as a controller to tell the robot how it should react in a situation. On the other hand, neural networks have broader applications in robot navigation problems. That includes landmark recognition, path-planning, and controller. Kohonen network is more popular in topological environment maps building. ALICE robot [21] recognized places where it had been before, using Kohonen's neural network clustering techniques. Tani [22] combined Kohonen and feedforward networks to navigate YAMABICO robot to a predetermined goal in arbitrary workspace without global information. Janet [23] was trying to globally self-localize a mobile robot utilizing Kohonen neural network as well. Using a topologically organized neural network of a Hopfield type, Glasins [24] demonstrated that his simulated robot could move from any arbitrary start position to any static or moving target position by avoiding both static and moving obstacles. There are many other related works involved in robot autonomous navigation. Nagata [25] had his cops and thieves robots controlled by a structured hierarchical neural network in the early 1990s. The success of ALVINN, MANIAC [26], and NEURO-NAV [27] also had an impact, especially their use of vision system to extract landmark recognition with neural networks. Meng had more involvement in working on the three mentioned issues. His navigation system included landmark detector, path planner, and rule-based supervisory controller.

In this section, we specifically select the indoor environment to perform two navigation behaviors using sonar sensor. No feature extraction is concerned due to the fact that the major purpose is to show that the proposed NiF-T is general enough for various feasibilities of control problems. The wall following behavior seems different from the hall centering behavior. Actually, both are required to lock onto their specified paths. From the control aspect, they are different mostly in the fact that wall following behavior needs only one side wall information.

For these two behaviors, some similar works have been done, such as [8], and [11]–[13], [28]. Cheng [8] had a simple network which minimized both orientation error and road offset. He showed that it was easier to find an optimal parameter set for the neuromorphic controller than for the proportional controller. Lubin [11] and Yu [12] similarly were interested in locking their robots onto the center lane. Lubin's network required thousands of training samples to train for thousands of iterations. Yu's incremental learning suffered from long training time by trial and error, as well. Truck backer-upper systems reported in [13] are the related work, which minimize both orientation error and position offset so that the truck will align with the desired loading dock. As many as 35 rules were developed in backup-truck fuzzy system and more than 3000 training samples were used to emulate the neural network (or with 35 training-sample vectors to train for 100 000 iterations). Holder's robot hugged wall [28]; however, the road offset could be more than an inch although the robot speed was very slow. Since these kinds of navigation behaviors are easily understood, incremental learning may not be necessary. Therefore, with capabilities of learning with minimal number of training rules, the proposed NiF-T may again be proved to be efficient in design and better in results.

C. Experimental Testbed

The mobile robot can be seen in Fig. 4; Small Mobile Autonomous Robotic-Testbed (SMAR-T) [28], is utilized to accomplish the robot behaviors discussed in this section. It has distributed multiprocessors which allow the high level control consisting of C/C++ routines to provide a multitude of functions. It is capable of intercommunicating with additional robots over wireless modems. Only one foot high and 18 in in diameter at its base, SMAR-T weights approximately 35 lbs. Its drive wheels are driven with a $2:1$ gear ratio to reach motor torque of 250 oz $\cdot$ in. The maximum speed it can reach is 1 ft/s.

D. Wall Following Behavior

1) The Control Scheme: The robot can be designated to follow at any specified distance from the right or left wall. When following the right wall, the robot sonar sensor scans only the right plane region (from -15.0° to $+90.0^\circ$ for the right plane and from $+15.0^\circ$ to -90.0° for the left plane). Unfortunately, due to hardware constraints, sufficient sensor information cannot be acquired for the negative region of the right plane and for the positive region of the left plane. The robot scanning regions and the input parameter denotations are shown in Fig. 2. We consider only two input parameters, road offset and orientation error, in this design. The robot speed can be one of the input parameters as well; for simplicity, it is set constant for the time being.

The membership functions are constructed as in Fig. 3. When the robot is in position between the wall and the specified distance, the road offset is in the negative region (d^-); otherwise, it is in d^+. The robot heading direction is supposed to align parallel with the wall. If the robot is heading toward the d^- region, the orientation error is a negative value; otherwise, it is positive. Since the sonar sensor can echo a distance as great as 30 ft, the road offset membership function is extended to 30 ft.

The control goal here is to align the robot parallel with the wall at a specified distance. For instance, when the robot is heading farther away from the specified distance trail toward the wall, the controller is supposed to turn the robot with a large positive degree. A set of parent rules is listed in Table I.

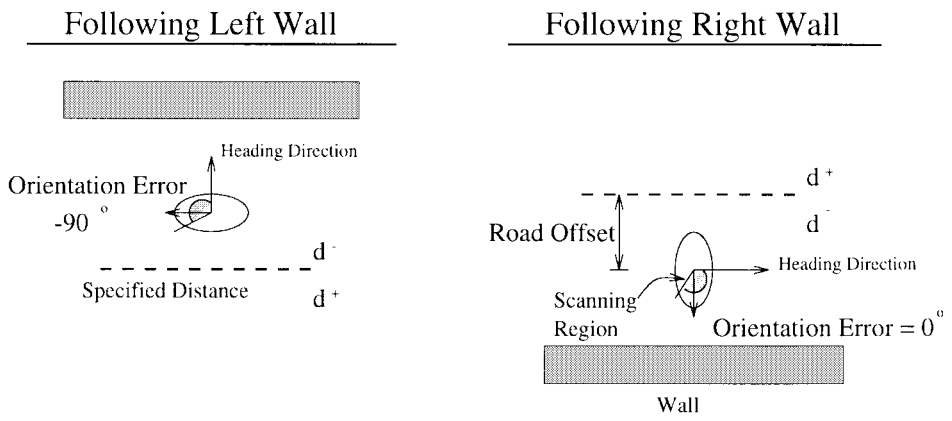


Fig. 2. The wall following scanning regions and input parameters.

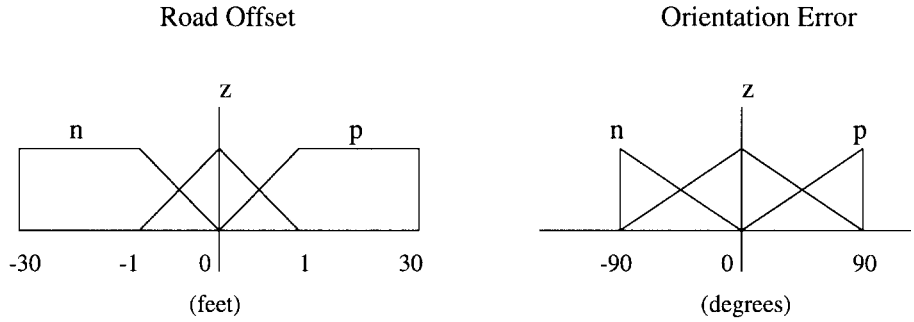
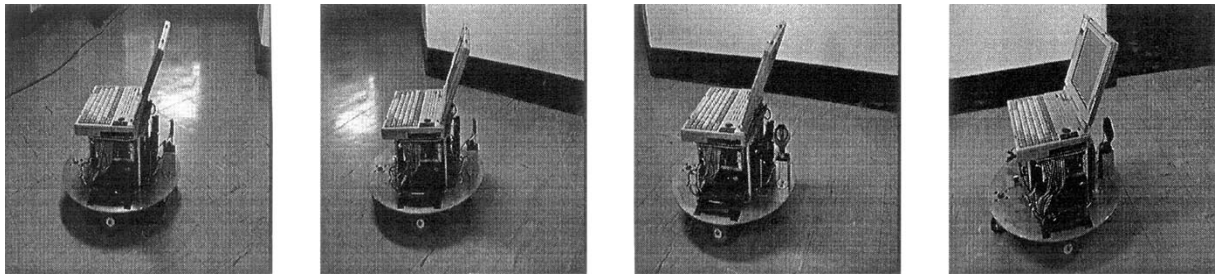


Fig. 3. The membership functions for wall following behavior.



(a) (b) (c) (d)

Fig. 4. SMAR-T successfully following the left wall in a hallway.

The weighting factors of the RNN output nodes are $-60.0°$, $0.0°$, and $60.0°$ in the order of ($-$l, z, l). We trained RNN and ORNN with the same five hidden layer nodes with the parameters of $\eta = 0.3$, $\lambda = 0.9$, and the desired rms error $=$ 0.05. The RNN converged at 966 iterations, while the ORNN converged at 90 iterations. This shows that the ORNN has a faster learning rate. We also include road offset and orientation error in the objective function; however, more emphasis has been placed on the road offset.

2) Experimental Validation: The experiments were carried out in a hallway with four doors on the left and three on the right. The doors were all closed and were recessed about four inches into the wall. Fig. 4 shows the robot, SMAR-T, following the left wall in the hallway. SMAR-T was at first a few feet from the left wall, facing away. The robot turned toward the wall and hugged at the specified distance. There are four sets of experiments for four different robot speeds from 50Hz (0.067 ft/s) to 300 Hz (0.4 ft/s). Every set includes ORNN with or without on-line learning. All were able to follow the wall at the specified distance. The results are shown in Fig. 5.

It is expected that the robot approaches its goal more steadily with slower robot speed [Fig. 5(a), (b), (e), and (f)] than with faster robot speed [Fig. 5(c), (d), (g), and (h)]. When the robot runs too fast, the input data acquired by the rotary sonar sensor are quickly outdated. In addition, the effect of learning is not noticeable, for two primary reasons: 1) the developed rules are good enough; and 2) there are too few iterations (the hallway is not long enough) to effectively learn whether there is any error in the rules. However, the errors that cause the robot to run oscillatory result more from hardware imperfection and the unstructured environment than from the controller itself. The robot, which can scan only $+15.0°$ for the left wall (more negative orientation error resolution when

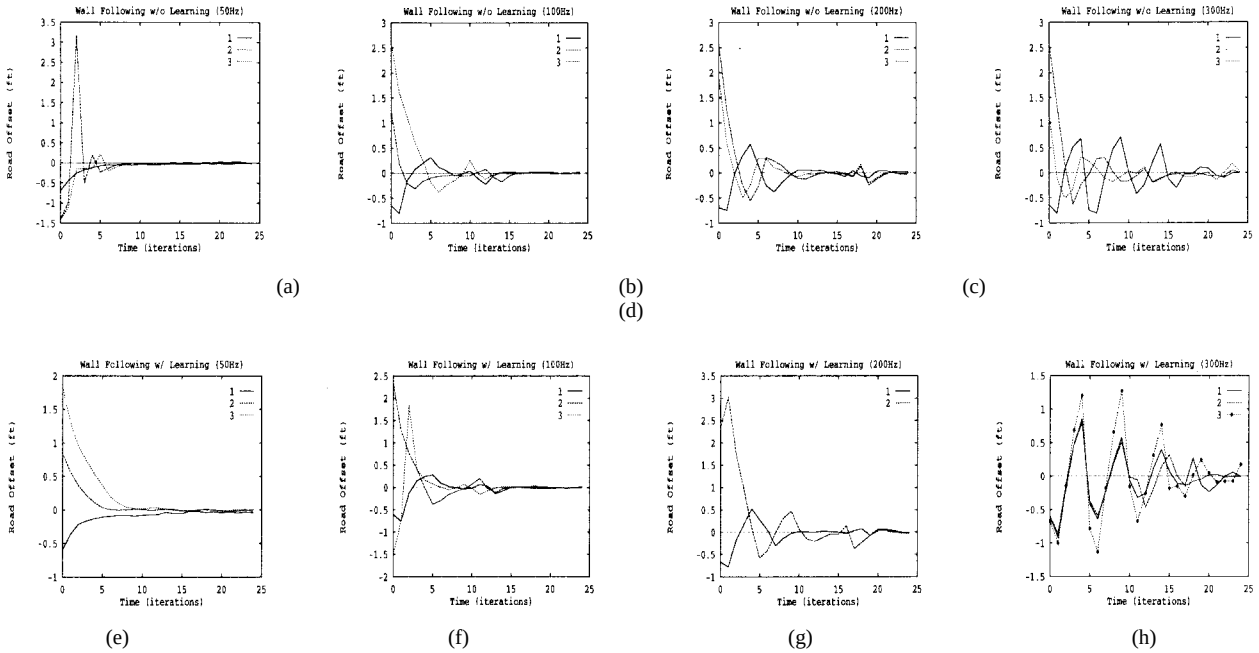


Fig. 5. The experimental results of wall following. The robot speed is set at 50, 100, 200, and 300 Hz from (a)–(d) (without on-line learning) and from (e)–(h) (with on-line learning). 300 Hz is equivalent to 0.4 ft/s.

hugging the left wall) and $-15.0°$ for the right wall, could produce bad sensory results, such as the noisy data shown on path 2 going from -1.5 to 3.0 in Fig. 5(a) and the path 3 going from -1.5 to around 2.0 in Fig. 5(f). The robot sensor information may mislead the robot that the orientation error is still negative while it is already in the positive road offset zone when hugging the left wall. Therefore, the third rule in Table I is continuously misfired until a positive orientation error is sensed, and the second rule sends the robot back to the right track. The unstructured environment (door depth) could produce some sudden errors, too. The obvious errors at around 17 iteration in Fig. 5(c) and (g) are due to the door depth. As we have mentioned before, when the ORNN learning takes place, the objective function's error constants may not be set too large. Otherwise, the controller may learn wildly (too fast), such as path 3 in Fig. 5(h).

E. Hall Centering Behavior

1) The Control Scheme: As we have mentioned in Section III-B, the hall centering problem is different from the wall following behavior mainly in the input data acquisition. The hall centering problem requires two side wall distances to determine the hall center point (see Fig. 6). Road offset, orientation error, and robot speed are the three input parameters. The robot speed is included this time so that its performance can be compared to the wall following behavior, which does not consider the robot speed as an input factor.

Fig. 7 shows the membership functions. The right side of the hall center is considered the positive region. The robot heading direction should eventually align with both right and left walls. Similarly, if the robot is heading toward the d^- region, the orientation error is a negative value. Due to the hardware constraints of SMAR-T, only a $\pm 15°$ of orientation error can be measured. SMAR-T speed can be set to a maximum of 300

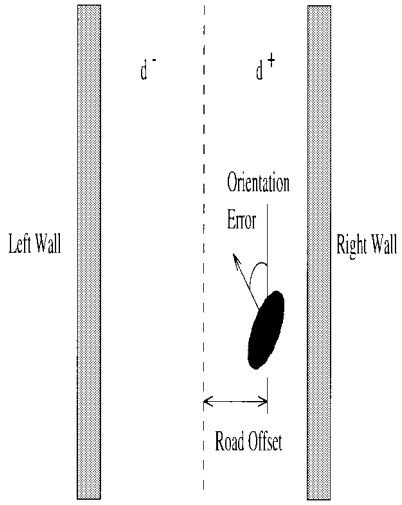


Fig. 6. Input parameters for the hall centering behavior.

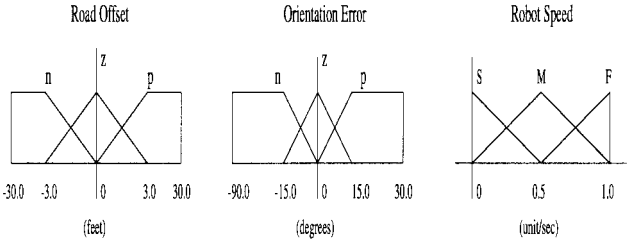


Fig. 7. Membership functions for hall centering.

Hz, and the robot will still stay reliably functional. Thus, the unit of one in the speed membership function corresponds to 300 Hz (about 0.4 ft/s).

The robot is supposed to lock onto the central lane. When the robot approaches the center slowly from the left side, the

TABLE III
THE NINE HALL CENTERING PARENT RULES

Rule	*Inputs*			*Output*	*Cond.*
	Following Offset	*Follower Speed*	*Leader Speed*		
1	z	z	S	z	Stable
2	p	p	F	-l	Critical
3	p	p	S	-m	
4	p	n	F	s	
5	p	n	S	z	
6	n	p	F	-s	
7	n	p	S	z	
8	n	n	F	l	
9	n	n	S	m	

controller may not react as fast to turn the robot to its left as when the robot speed is fast. Nine parent rules are listed in Table III. The weighting factors assigned to the RNN output nodes are $\pm 30.0(\pm l)$, $\pm 20.0(\pm m)$, $\pm 5.0(\pm s)$, and 0.0(z)°. We trained RNN and ORNN with the same five hidden layer nodes with the parameters of $\eta = 0.3$, $\lambda = 0.9$, and the desired rms error $= 0.05$. The RNN converged at 341 iterations, while the ORNN converged at 84 iterations. This again shows that the ORNN has a faster learning rate, since the network is typically smaller. Road offset and orientation error are the parameters included in the objective function. However, we placed more emphasis on correcting the road offset.

2) Experimental Validation: The experiments here are again conducted in the same environment as in Section III-D2. There are three sets of experiments with the robot speed of 50, 100, and 200 Hz, respectively. Most of the runs approached around the hall center. In general, the controller shows better results with the help of the ORNN on-line learning capability. Fig. 8 is SMAR-T moving toward the hall center.

In Fig. 9(a), the robot on paths 4 and 5 failed to return to the hall center. The principal reason could be that not enough weight is assigned to the output label $\pm m$ ($\pm 20°$). Therefore, rule 9 in Table III is not able to turn the robot back to the center in that situation. However, with on-line learning, the robot almost locked onto the center path in all five runs [Fig. 9(b)]. Except for the fact that for the second path, the error constant of the objective function was set a little bit too high (-0.01), the path appears to have a higher overshoot. The same occurs for the learning part of Fig. 9(d) path 2 and of (f) path 2. Some large number jumps in Fig. 9(a) path 4 and (f) path 2 are caused by the sensor noises. Still, when the robot is operating at high speed with the limited sonar sensor scanning regions, the sensor uncertainty make the robot hard to lock onto the center [plots (e) and (f)].

IV. MULTIROBOT CONVOYING USING NiF-T

Multiple mobile robot convoying is especially difficult to obtain their mathematical models, although it is common and easy for human beings to perform this task, such as in our everyday driving experience. Not only convoying behavior is important in the IVHS application, it is also useful in the multiple robot rescue mission. For example, if a robot is malfunctional in a clutter, hazardous environment, a leader may guide the robot out using the convoying behavior. Due to its high nonlinearity, we hope that the NiF-T that has nonclassical means would accomplish the task more efficiently.

A. Problem Statement

For this convoying behavior, a leader is neither cross communicating with nor giving any explicit hint to the follower. The follower is supposed to use its sensing information to change its speed, moving direction, and orientation. The following behaviors are on the part of the follower.

1) Keep close to the specified distance away from the leader when moving.
2) Stop at the specified distance away from the leader when the leader stops.
3) Change speed smoothly and instantly, corresponding to the leader's speed change.
4) Move away with the behaviors described in 1–3 if the leader is moving toward the follower (herding behavior).

B. Review of Other Approaches

No matter how capable a single robot is, multiple-robot systems can economically and efficiently accomplish complex tasks that no single robot can accomplish. Instead of building a single powerful robot for each separate task, using several simple robots can be easier, cheaper, more flexible, and more fault tolerant [29]. Cooperative behaviors can be carried out by a team of robots (usually mobile robots), either with different skills (referred to as heterogeneous robots) or with the same skills (referred to as homogeneous robots). The applications that have been worked on can be classified mainly into three types: cooperative transportation, cooperative sensing, and foraging [30]. In cooperative transportation, multiple mobile robots transport objects such as a box/furniture [31], [32] cooperatively. Cooperative sensing makes individual robots that are equipped with their own sensors recognize their own environment and put their pieces of information together into a complete picture. Communication is a key design issue for multiple-robot systems. Different communication frameworks were discussed in [33]. There are researchers proposing robot architecture such as [34], while others are putting their efforts toward solving geometric problems [35], [36]. However, very few applications of cooperative robotics have been yet reported, and supporting theory is still in its formative stages [37].

Convoying behavior involves the use of multiple robots. Wang [38] has simulated some navigation path-planning strategies for more than three mobile robots to move in formation. Sensing and communication for robot convoy were experimented in [39]. The work discussed in this section is focusing on the motion control using the proposed NiF-T. Reference [40] has implemented fuzzy control on this similar problem. As many as 36 rules were derived in that work, but no herding behavior can be performed, and the robot does not convoy at a specified distance. With the methods described here, fewer rules are required. In addition, without any communication and hint from the leader, the follower is convoying at the specified safe distance away from the leader.

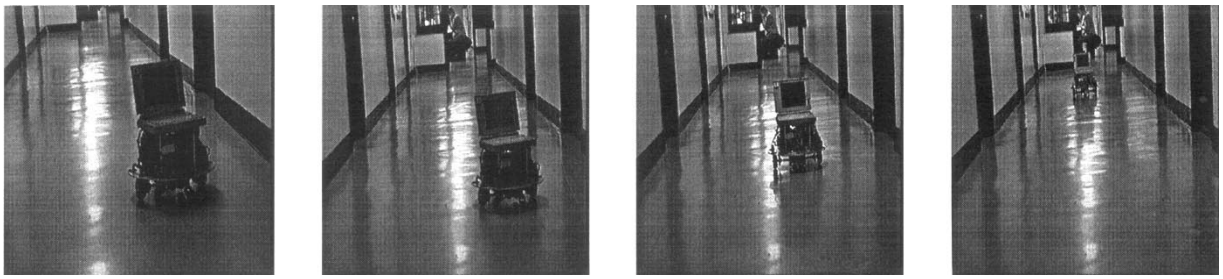


Fig. 8. SMAR-T was successfully moving to the hallway center.

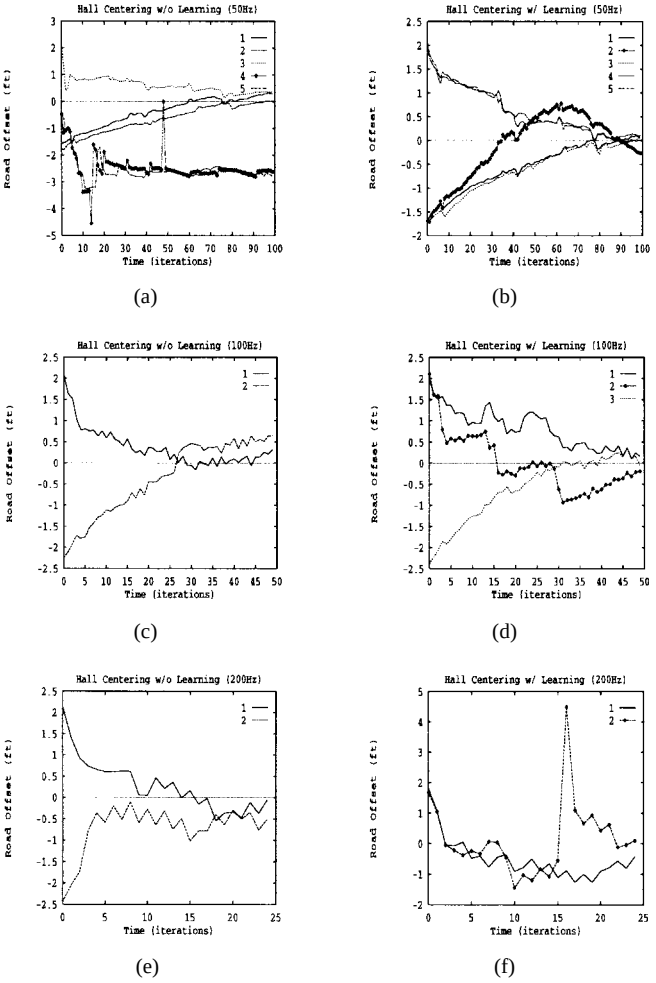


Fig. 9. The experimental results of hall centering.

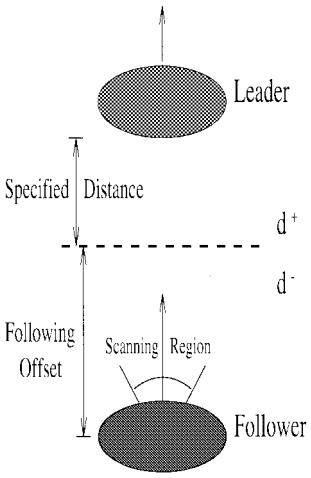


Fig. 10. Specifications for the multirobot convoying behavior.

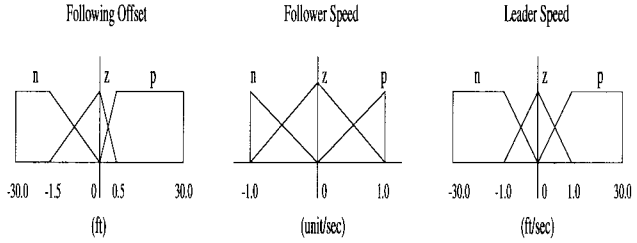


Fig. 11. The membership functions of robot convoy.

C. Experimental Testbed

SMAR-T and ELVIS (Fig. 12) are the robots to be used to demonstrate convoying behavior. ELVIS is just an arbitrary wheeled mobile robot to be utilized as a leader regardless of its other real-time processing capabilities. SMAR-T is the robot which has the NiF-T to coordinate its motion with ELVIS.

D. The Control Scheme

While convoying, four basic parameters are constantly monitored. Without any communication from the leader, the follower has to measure its distance from the leader, know its own and the leader's speed, and detect the leader's orientation. Using sonar sensor, the distances can easily be obtained. Subtracting the two immediate distances gives us the relative speed of the two robots. Since the follower has the record of its own speed, adding that speed to the relative speed results in the leader speed. The follower is always trying to align itself with the closest point to the leader. Although both the speed and the steering angle of the follower are being controlled continuously, we are currently more interested in designing the speed controller using the NiF-T.

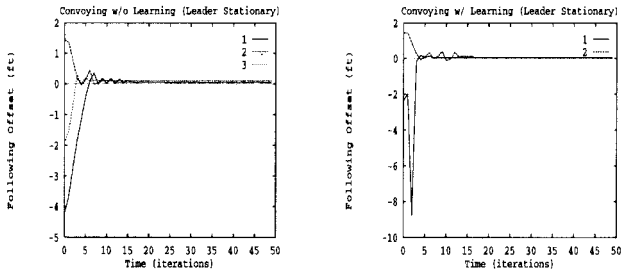
Three parameters—following offset, follower speed, and leader speed—are the inputs of the controller. The follower is considered to be in the negative following offset region (d^-) when it is more than the specified distance from the leader (see Fig. 10). When the robots go forward, the speed is positive; otherwise, the speed is negative. The constructed membership functions appear in Fig. 11. SMAR-T speed can be set to a maximum of 300 Hz and the robot will still stay reliably functional. Thus, the unit of one in the speed membership function corresponds to 300 Hz (about 0.4 ft/s).

(a) (b) (c) (d)

Fig. 12. Experimental demonstration of SMAR-T successfully convoying ELVIS in a hallway.

TABLE IV
THE PARENT RULES OF ROBOT CONVOY

Rule	Inputs			Output	Cond.
	Following Offset	Follower Speed	Leader Speed		
1	z	z	z	z	Stable
2	p	p	p	-m	Critical
3	p	p	n	-l	
4	p	n	p	z	
5	p	n	n	-m	
6	n	p	p	m	
7	n	p	n	z	
8	n	n	p	l	
9	n	n	n	m	



(a) (b)

Fig. 13. Results of convoying experiments where the leader stayed stationary. The follower started from different initial positions, and it was able to stop at the specified "safe" distance in a very short time.

The parent rule set developed in Table IV is typically based on observation of daily driving behavior: the greater the distance between my car and the car ahead of me, and the faster that car is traveling, the harder I will tend to accelerate. I will also stop at a safe distance from the front car at a red light. If a car is backing toward me and passes the specified "safe" distance, that will intrude on my sense of security and make me backing up as well. Although backing up and the exact "safe" distance specification are not really necessary in our daily driving practice, they just show that the NiF-T can easily assimilate many interesting behaviors. The weighting factors of the RNN output nodes are $-1.75(-\text{l})$, $1.0(-\text{m})$, 0.0(z), 1.0(m), and 1.5(l). We trained RNN and ORNN with the same five hidden layer nodes with the parameters of $\eta = 0.3$, $\lambda = 0.9$, and the desired rms error $= 0.05$. The RNN converged at 735 iterations, while the ORNN converged at 89 iterations. This again shows that the ORNN has faster learning rate. The following offset is only to minimize in the objective function.

E. Experimental Validation

Extensive set of real-world experimental studies were undertaken to validate the unique features of NiF-T architecture in realizing multirobot convoying. Two sets of experiments are discussed here. First, when the leader is stationary, the follower starts from any arbitrary initial position to reach the specified safe distance and stop (Fig. 12). Secondly, as the leader keeps moving randomly, the follower tries to keep up with the change. Learning is unnecessary or undesirable in the second case because the leader's motion is irregular or unformatted. In both cases, the follower convoyed the leader closely and smoothly by varying its speed and changing its direction. When the leader stopped, the follower stopped very close to the specified safe distance. The follower backed up quickly when the leader moved backward toward it.

In Fig. 13, the leader was stationary. The follower was initially placed in either the negative or positive following region. It stopped very close to the specified safe distance in a very short period of time, with or without learning. In plot (b) path 1, the following offset, which abruptly drops to -8 (ft), is the sensor noise.

The follower, SMAR-T, cannot acquire its own speed directly from the hardware; therefore, the speed recording must be done in the software. However, SMAR-T does not response to any frequency that is set below 30.7 Hz. As a result, the calculated leader speed may not be exact. This is obvious in Fig. 14(a) and (b) (following offset was measured in feet; leader speed was in feet per second). The leader actually was moving forward all the time; yet the plots show that the leader sometimes moved backward. The follower was still able to convoy closely with the leader although the sensors were noisy and uncertain. When the leader makes a sudden change in speed, the follower may lag or exceed the specified following safe distance for a while. Generally within ten iterations, the follower catches up with the leader so long as the leader does not go faster than the maximum driving speed of SMAR-T (0.4 ft/s in this case).

V. CONCLUDING REMARKS

A Neural integrated Fuzzy conTroller (NiF-T), which integrates the fuzzy logic representation of human knowledge with the learning capability of neural networks, is developed for

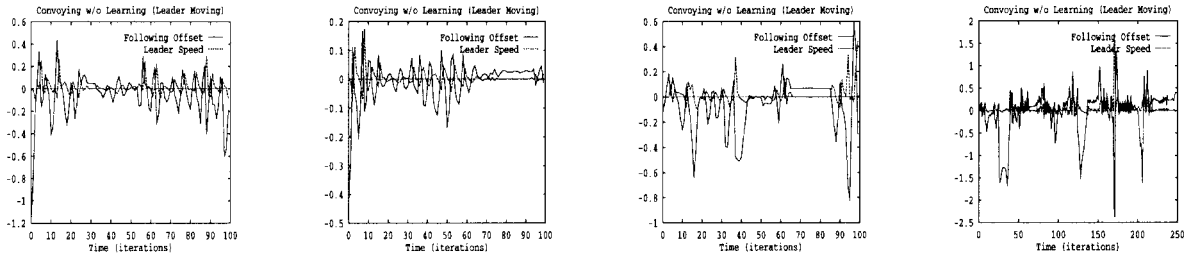


Fig. 14. Results of convoying experiments where the follower convoyed the moving leader. The leader kept changing its speed and direction. The follower was able to cope with the change and stop close to the specified safe distance when the leader stopped.

nonlinear dynamic control problems. With the help of fuzzy logic, numerical sample data are no longer needed for the network training. Instead, the reliable expert rules are used to train the networks. In addition, membership functions provide continuous input representations. They make the controller less sensitive to slight variations in the physical parameters and control goal. Likewise, with the help of neural networks, fewer fuzzy rules are to be extracted from the human experts and the control action can be fine-tuned on-line. The NiF-T can also endure highly noisy signals.

In this paper, the NiF-T has been applied to control the robot motion—steering angle, heading direction, and speed. Only five rules were used to train the wall following NiF-T, while nine were used for the hall centering. Compared with previous works which require many rules, even thousands of training data and iterations, this can be considered an accomplishment. With learning capability, robot behaviors can be modified. If the parameters (especially the objective function constants) are properly set, learning will only enhance a controller's performance. When a behavior (wall following) is already under control, the learning does not have a negative effect on the results. However, bad results of the hall centering behavior were improved by the NiF-T. Additionally, with fewer rules developed than [40]'s work, the convoying here shows more interesting behaviors. The follower stopped at the specified distance (within an inch) in a short length of time. It followed the leader closely with smooth and instant speed change. When the leader moved backward toward the follower, the follower instantly backed up as well. When the leader makes a sudden change in speed or direction, the follower may lag or exceed the specified following safe distance for a while. Generally within 10 iterations, the follower catches up with the leader so long as the leader does not go faster than the maximum driving speed of SMAR-T (0.4 ft/s in this case). For all of the described behaviors—wall following, hall centering, and convoying, their RNN's are trained only for a few hundred iterations and so are their ORNN's trained for only less than one hundred iterations to learn their parent rule sets.

Finally, we would like to comment about the generality of NiF-T. We have used this architecture to control a difficult nonlinear dynamic problem, that of BBB [41]. This required real-time feedback and an ability to handle noisy sensory signals associated with uncertainty. The performances with NiF-T was significantly better than that achieved using only fuzzy logic control [42]. Additionally, we have implemented a modular version of NiF-T for race car navigation [43]. MoNiF shows better racing skill than a neural controller and a rule-based controller.

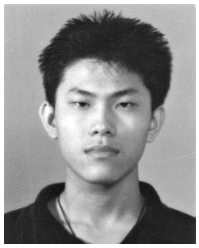
ACKNOWLEDGMENT

The authors would like to thank Prof. J. C. Bezdek for his valuable and insightful comments.

REFERENCES

[1] W. Pedrycz and A. F. Rocha, "Fuzzy-set based models of neurons and knowledge-based networks," *IEEE Trans. Fuzzy Syst.*, vol. 1, Nov. 1993.
[2] S. Horikawa, T. Furuhashi, and Y. Uchikawa, "On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm," *IEEE Trans. Neural Networks*, vol. 3, pp. 801–806, Sept. 1992.
[3] M. Funabashi, A. Maeda, Y. Morooka, and K. Mori, "Fuzzy and neural hybrid expert systems: Synergetic AI," *IEEE Expert*, vol. 10, pp. 32–40, Aug. 1995.
[4] C. M. Higgins and R. M. Goodman, "Fuzzy rule-based networks for control," *IEEE Trans. Fuzzy Syst.*, vol. 2, Feb. 1994.
[5] H. Ishibuchi and H. Tanaka, "Neural networks that learn from fuzzy if-then rules," *IEEE Trans. Fuzzy Syst.*, May 1993.
[6] F. Bouslama and A. Ichikawa, "Application of neural networks to fuzzy control," *Neural Networks*, vol. 6, pp. 791–799, 1993.
[7] H. R. Berenji and P. Khedkar, "Learning and tuning fuzzy logic controllers through reinforcements," *IEEE Trans. Neural Networks*, vol. 3, pp. 724–740, Sept. 1992.
[8] R. M. H. Cheng, J. W. Xiao, and S. LeQuoc, "Neuromorphic controller for AGV steering," in *ICRA*, Nice, France, May 1992.
[9] M. Chiaberge and L. M. Reyneri, "Cintia: A neuro-fuzzy real-time controller for low-power embedded systems," *IEEE Micro*, pp. 40–47, June 1995.
[10] S. Thaler, "Fuzzy rule generation based on a neural network approach," *Electron. Eng.*, pp. 43–50, July 1993.
[11] J. M. Lubin, E. C. Huber, S. A. Gilbert, and A. L. Kornhauser, *Analysis of a Neural Network Lateral Controller for an Autonomous Road Vehicle*. Warrendale, PA: Soc. Auto. Eng., 1992, pp. 23–44.
[12] G. Yu and I. K. Sethi, "Road following using incremental learning," Tech. Rep., Dept. Comput. Sci., Wayne State Univ., Detroit, MI, 1994.
[13] B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamical Approach to Machine Intelligence*. Englewood Cliffs, NJ: Prentice-Hall, 1991.
[14] J. J. Leonard and H. F. D. Whyte, *Directed Sonar Sensing for Mobile Robot Navigation*. Norwell, MA: Kluwer, 1992.
[15] C. Chen and M. M. Trivedi, "Reactive locomotion control of a tracked mobile manipulator," in *IROS*, Yokohama, Japan, July 1993, pp. 1349–1356.
[16] M. M. Trivedi, "The abc of intelligent robotic systems," in *Proc. Automatique et Robotique, DRET*, Paris, France, Jan. 1993.
[17] ———, *Encyclopedia of Science and Technology*. New York: McGraw-Hill, 1994, pp. 226–229.
[18] H. Surmann, J. Huser, and L. Peters, "A fuzzy system for indoor mobile robot navigation," in *FUZZ-IEEE*, Yokohama, Japan, 1995.
[19] 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.
[20] T. Fukuda, S. Ito, F. Arai, and Y. Yokoyama, "Navigation system based on ceiling landmark recognition for autonomous mobile robot," in *IROS*, Pittsburgh, PA, Aug. 1995, vol. 2, pp. 150–155,

[21] U. R. Zimmer and E. V. Puttkamer, "Realtime learning on an autonomous mobile robot with neural networks," in *Euromicro 94-Realtime Workshop*, Vaesteraas, Sweden, June 1994, pp. 15–17.
[22] J. Tani and N. Fukumura, "Learning goal-oriented navigation as attractor dynamics for a sensory motor system," in *Proc. 1993 Int. Joint Conf. Neural Networks*, pp. 1747–1752.
[23] J. A. Janet *et al.*, "Global self-localization for autonomous mobile robots using self-organization Kohonen neural networks," in *IROS*, Pittsburgh, PA, Aug. 1995, vol. 3, pp. 504–509.
[24] R. Glasins, A. Komoda, and S. C. A. M. Gielen, "Neural network dynamics for path planning and obstacle avoidance," *Neural Networks*, vol. 8, no. 1, pp. 125–133, 1995.
[25] S. Nagata, M. Sekiguchi, and K. Asakawa, "Mobile robot control by a structured hierarchical neural network," *Contr. Syst.*, vol. 10, pp. 69–76, Apr. 1990.
[26] T. M. Jochem, D. A. Poonerlean, and C. E. Thorpe, "Maniac: A next generation neurally based autonomous road follower," in *Int. Conf. Intelligent Autonomous Systems*, Pittsburgh, PA, 1993.
[27] M. Meng and A. C. Kak, "Neuro-nav: A neural network based architecture for vision-guided mobile robot navigation using nonmetrical models of the environment," in *ICRA*, Atlanta, GA, May 1993, pp. 750–757.
[28] M. B. Holder, "Design and implementation of an integrated multiprocessor mobile robot for experimental research in cooperative robotics," M.S. thesis, Univ. Tennessee, Knoxville, Dec. 1994.
[29] M. M. Trivedi, "Intelligent robots: Control and cooperation," in *Proc. SPIE Applications Fuzzy Logic Technology II*, Orlando, FL, Apr. 1995, vol. 2493.
[30] L. E. Parker, "The effect of action recognition and robot awareness in cooperative robotic teams," in *IROS*, Pittsburgh, PA, Aug. 1995, vol. 1, pp. 212–219.
[31] D. Rus, B. Donald, and J. Jennings, "Moving furniture with teams of autonomous robots," in *IROS*, Pittsburgh, PA, Aug. 1995, vol. 1, pp. 235–242.
[32] M. J. Mataric, M. Nilsson, and K. T. Simsarian, "Cooperative multirobot box-pushing," in *IROS*, Pittsburgh, PA, Aug. 1995, vol. 3, pp. 556–561.
[33] H. Asama, M. K. Habib, and I. Endo, "Functional distribution among multiple mobile robots in an autonomous and decentralized robot system," in *ICRA*, Sacramento, CA, Apr. 1991, pp. 1921–1926.
[34] F. R. Noreils, "Toward a robot architecture integrating cooperation between mobile robots: Application to indoor environment," *Int. J. Robot. Res.*, vol. 12, pp. 79–98, Feb. 1993.
[35] D. Kurabayashi, J. Ota, T. Arai, and E. Yoshida, "An algorithm of dividing a work area to multiple mobile robots," in *IROS*, Pittsburgh, PA, Aug. 1995, pp. 286–291.
[36] C.-F. Lin and W.-H. Tsai, "Optimal assignment of robot tasks with precedence for multi-robot coordination by disjunctive graphs and state-space search," *J. Robot. Syst.*, vol. 12, no. 4, pp. 219–236, 1995.
[37] Y. U. Cao, A. S. Fukunaga, A. B. Kahng, and F. Meng, "Cooperative mobile robotics: Antecedents and directions," in *IROS*, Pittsburgh, PA, Aug. 1995, vol. 1, pp. 226–234.
[38] P. K. C. Wang, "Navigation strategies for multiple autonomous mobile robots moving in formation," *J. Robot. Syst.*, vol. 8, no. 2, pp. 177–195, 1991.
[39] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "Experiments in sensing and communication for robot convoy," in *IROS*, Pittsburgh, PA, Aug. 1995, pp. 268–273.
[40] S. B. Marapane, M. B. Holder, and M. M. Trivedi, "Motion control of cooperative robotic teams through visual observation and fuzzy logic control," in *ICRA*, Minneapolis, MN, Apr. 1996.
[41] K. C. Ng and M. M. Trivedi, "Neural integrated fuzzy controller (NIFT) and real-time implementation of a ball balancing beam (BBB)," in *ICRA*, Minneapolis, MN, Apr. 1996, vol. 2, pp. 1590–1595.
[42] ———, "Fuzzy logic controller and real-time implementation of a ball balancing beam," in *Proc. SPIE, Applications Fuzzy Logic Technology II*, Orlando, FL, Apr. 1995, vol. 2760, pp. 261–272.
[43] K. C. Ng, R. Scorcioni, M. M. Trivedi, and N. Lassiter, "Monif: A modular neuro-fuzzy controller for race car navigation," in *Proc. IEEE Computational Intelligence Robotics Automation*, Monterey, CA, July 1997.



Kim Chai Ng (S'94–M'95) was born in Lima Kedai, Malaysia. He received the B.S.E.E. and M.S.E.E. degrees in 1993 and 1995, respectively, from the University of Tennessee, Knoxville, where he earned the Top Graduating Senior in the College of Engineering for the B.S.E.E. degree. He is currently pursuing the Ph.D. degree at the University of California, San Diego.

His research interests are in intelligent systems and machine vision, specifically in novel and efficient mechanisms for 3-D information extraction.

Mohan Manubhai Trivedi (S'76–M'79–SM'86) is a Professor in the Department of Electrical and Computer Engineering, University of California, San Diego, where he also serves as the Director of the Program in Advanced Manufacturing. He and his team are engaged in a broad range of sponsored research studies in active perception and machine vision, integrated intelligent systems, and interactive graphical interfaces for human–machine systems. He serves as the Editor-in-Chief of *Machine Vision and Applications*. He is a frequent consultant to various national and international industry and government agencies.

Dr. Trivedi received the Pioneer Award (Technical Activities) and the Meritorious Service Award from the IEEE Computer Society and the Distinguished Alumnus Award from Utah State University, Logan. He is a Fellow of the International Society for Optical Engineering (SPIE).