

# Use of Neural Fuzzy Networks with Mixed Genetic/Gradient Algorithm in Automated Vehicle Control

Sunan Huang and Wei Ren, *Senior Member, IEEE*

**Abstract**—This paper is concerned with the design of automated vehicle guidance control. First, we propose to implement the guidance tasks using several individual controllers. Next, a neural fuzzy network (NFN) is used to build these controllers, where the NFN constructs are neural-network-based connectionist models. A two-phase hybrid learning algorithm which combines genetic and gradient algorithms is employed to identify the NFN weightings. Finally, simulations are given to show that the proposed technology can improve the speed of learning convergence and enhance the performance of vehicle control.

**Index Terms**—Acceleration control, autonomous vehicles, backpropagation, braking, fuzzy neural networks, genetic algorithms.

## I. INTRODUCTION

**T**RAFFIC congestion is a worldwide problem. Intelligent Vehicle/Highway Systems (IVHS's) have been proposed to help solve this problem [1]. Much work has been done on designing IVHS systems, which belongs to one of three categories: 1) hierarchical control system design for fully automated traffic, in which vehicles are all automated and are organized into platoons through interactive vehicle-to-vehicle communication [1]–[3]; 2) autonomous intelligent cruise control, i.e., keeping a desired velocity when there is no vehicle in front and keeping a desired spacing (possibly velocity dependent) when there is a vehicle in front [4]–[6]; and 3) autonomous vehicle guidance, which requires sensing, actuation, and computational “intelligence” on board the vehicle [7]–[10].

Specifically, we are interested in designing the system of the third category. Most studies in this area depend upon a precise input–output math model of the car “plant” and its environment [6], [7]–[9], [11]. However, the reactions of a driver to various traffic situations are perhaps not based on an exact mathematic relationship, but on a set of vague driving rules developed through experience. Recently, some authors used fuzzy set concepts to design automated driving systems, since fuzzy systems for handling real-world processes do not use an input–output math model or exact car parameters. In [12], a vehicle-following model that employs the fuzzy

inference system is proposed. It predicts the reaction of the driver by analyzing the fuzzy values of the leading vehicle's acceleration and the distance between the leading vehicle and the following vehicle. Furthermore, [13] extends the results of [12] to a fuzzy throttle and brake control for a smart car. However, fuzzy systems encounter different difficulties, such as how to determine the fuzzy logic rules and the membership functions. To solve these problems, neural fuzzy networks (NFN's) have been suggested in the literature for implementing fuzzy logic systems [14]–[17]. NFN's are multilayer connectionist models with improved learning capabilities that combine the benefits of neural networks and the fuzzy logic systems into a unified framework [17]. In particular, NFN's share the computational power of neural networks and the reasoning and the decision making of fuzzy systems. Hence, the NFN's constructs exhibit, generally, better performance for controlling plants and adapting to their environments as compared to the conventional fuzzy systems and neural networks. In [14] and [15], an NFN is used to implement a fuzzy inference system. The authors use the Takagi–Sugeno rule of inference with bell-shaped membership functions. The premise space is uniformly partitioned and all possible combinations of the linguistic sets are considered in the formulation of the rule base, thus fixing the structure of the NFN architecture. A hybrid learning scheme is employed to adjust the network parameters. In particular, the parameters of the rule output functions are updated using the least-squares algorithm, while the membership parameters are tuned by means of the backpropagation algorithm. In [18], a fuzzy neural network is developed to realize fuzzy rules with fuzzy consequents. A real-time training scheme is proposed that combines both structure and parameter learning. Structure learning, which is focused on the consequent side of the fuzzy inference system, decides the proper output partitions and the respective rule connections. This task is achieved by using the fuzzy similarity measure which indicates the degree to which two fuzzy sets are equal. The parameter learning task is performed using the backpropagation algorithm. The fuzzy neural construct is incorporated into a unified control architecture which uses the reinforcement learning technique to decide the proper control actions.

Essentially, the problem of controlling a complex system using NFN controllers can be considered as a multiparameters optimization problem. Since the NFN control systems are highly nonlinear, the choice of optimization technique may not be obvious and easy. Genetic algorithms (GA's) are a hope

Manuscript received April 7, 1997; revised December 17, 1998. Abstract published on the Internet August 20, 1999. This work was supported by the PATH Project, PATH MOU 160.

S. Huang was with the Electrical Engineering and Computer Sciences Department, University of California, Berkeley, CA 94703 USA. He is now with the Electrical Engineering Department, National University of Singapore, Singapore 119260.

W. Ren is with the Electrical Engineering and Computer Sciences Department, University of California, Berkeley, CA 94703 USA.

Publisher Item Identifier S 0278-0046(99)08474-9.

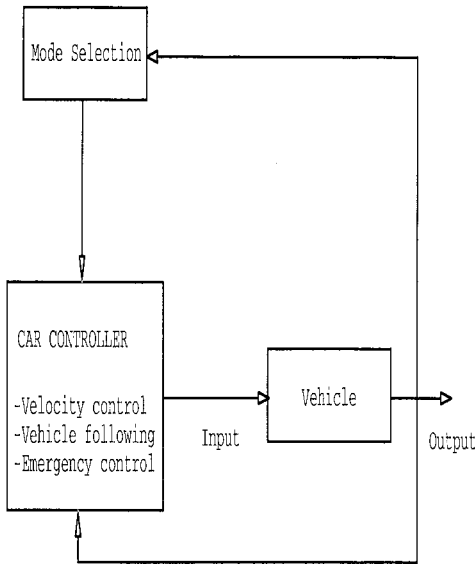


Fig. 1. Vehicle guidance control system.

for solving such problem. GA's are kinds of search algorithms based on the mechanics of nature genetics which are capable of rapidly locating near-optimal solutions to difficult problems [19]. They have been used to alter membership function in response to the changes of environment to produce more efficient fuzzy control performance [20], [21]. By introducing the NFN construct, the authors in [16] discuss an adaptive learning scheme of a fuzzy-filtered neural network. They provide three architectures for fuzzy-filtered neural networks, which employ one-dimensional fuzzy filters, two-dimensional fuzzy filters, and GA-based fuzzy filters. The effectiveness of the proposed models are verified by plasma spectrum analysis. In [22], a reinforcement learning technique is applied to a multilayer neural network model of a fuzzy logic controller. The proposed algorithm uses the GA through reinforcement learning architecture to learn fuzzy logic control rules, even when only weak information such as a binary target of "success" or "failure" signal is available.

In this paper, an NFN is considered for designing intelligent controllers of an autonomous vehicle guidance system. Fig. 1 shows how this guidance system performs closed-loop control of the vehicle. It includes mode selection and car controller. Our work focuses mainly on implementing the car controller that is composed of several individual NFN controllers. A two-stage learning scheme is employed in this paper for training NFN's. Simulations exhibit good learning capability and control performance of the proposed technology.

## II. VEHICLE DYNAMICS AND SENSOR SYSTEM

Vehicle models have been developed by many authors for a variety of purposes (see [23] and [26]). For a longitudinal control design, the system may be considered as a mathematical model (see [26])

$$\dot{x} = v \quad (1)$$

$$\dot{v} = -\frac{1}{m}(F - K_d v^2 + d_m) \quad (2)$$

where  $x$  is the longitudinal position of the vehicle,  $v$  is the velocity of the vehicle,  $m$  is the mass of the vehicle,  $K_d v^2$  specifies the force due to the air resistance,  $d_m$  gives mechanical drag forces, and  $F$  is the engine traction force, which we assume to be used to control the vehicle. The constants [26]  $K_d = 0.44$  kg/m,  $m = 916$  kg, and  $d_m = 352$  kg·m/s<sup>2</sup> are used in our paper.

We assume that each car has a combining radar and magnetometer system that provides the following capabilities.

- 1) It can detect velocity, acceleration, and distance measurements of the vehicle in front, as well as measurements of the vehicle's own velocity and acceleration.
- 2) It can detect the left-change-lane or right-change-lane light sent from a car in an adjacent lane.
- 3) It can detect vehicles in adjacent lanes. The detection ranges are taken to be the following constants:
  - a) in regions 1 and 2, 60 m;
  - b) in regions 3 and 4,  $\pm 30$  m from the center of the car.

With these assumptions in mind, an automated vehicle driving system should handle all possible traffic scenarios.

## III. INTELLIGENT AUTOMOTIVE GUIDANCE SYSTEM

In a typically envisioned driving scenario, the human driver first guides the vehicle into a highway traffic flow, and then sets the path of the vehicle, such as lane number and optimal speed of the individual vehicle, to the on-board computer. After this is completed, the driver presses a button to hand over control to the computer, and the vehicle starts automatic control by an intelligent guidance system. The guidance system should be able to perform three main tasks. The first is to track the immediately preceding car at a safe distance. The second task is to track an optimal velocity as much as possible without violating safety when no preceding cars are detected by the on-board sensors. The third task is that it should be able to handle the special controls, such as creating a space for a cut-in car and changing lane. To complete these tasks, the control system can be distinguished as two parts, as shown in Fig. 1, i.e., making a decision to obtain an appropriate control mode, and implementing precise control required by that mode.

The block diagram in Fig. 2 depicts the logic routes of the mode selection. At every iteration of the control loop, the mode selection analyzes sensor information to determine an appropriate control mode for the current vehicle, resulting in a longitudinal control (tracking optimal speed, tracking self-velocity, vehicle following, emergency) or a lateral control (changing lane). To help understand the decision-tree flow diagram, optional conditions in the ellipse block need to be explained clearly.

- *Check if car has completed lane change* is used to determine if controlled car has moved into a new lane.
- *Check if it is in its assigned lane* is used to determine if the vehicle is in its assigned lane, which was set in advance by the human driver. If *yes*, then the vehicle will search the next branch down in the decision tree. Otherwise, the vehicle will try to move to the adjacent lane closest to the assigned lane.

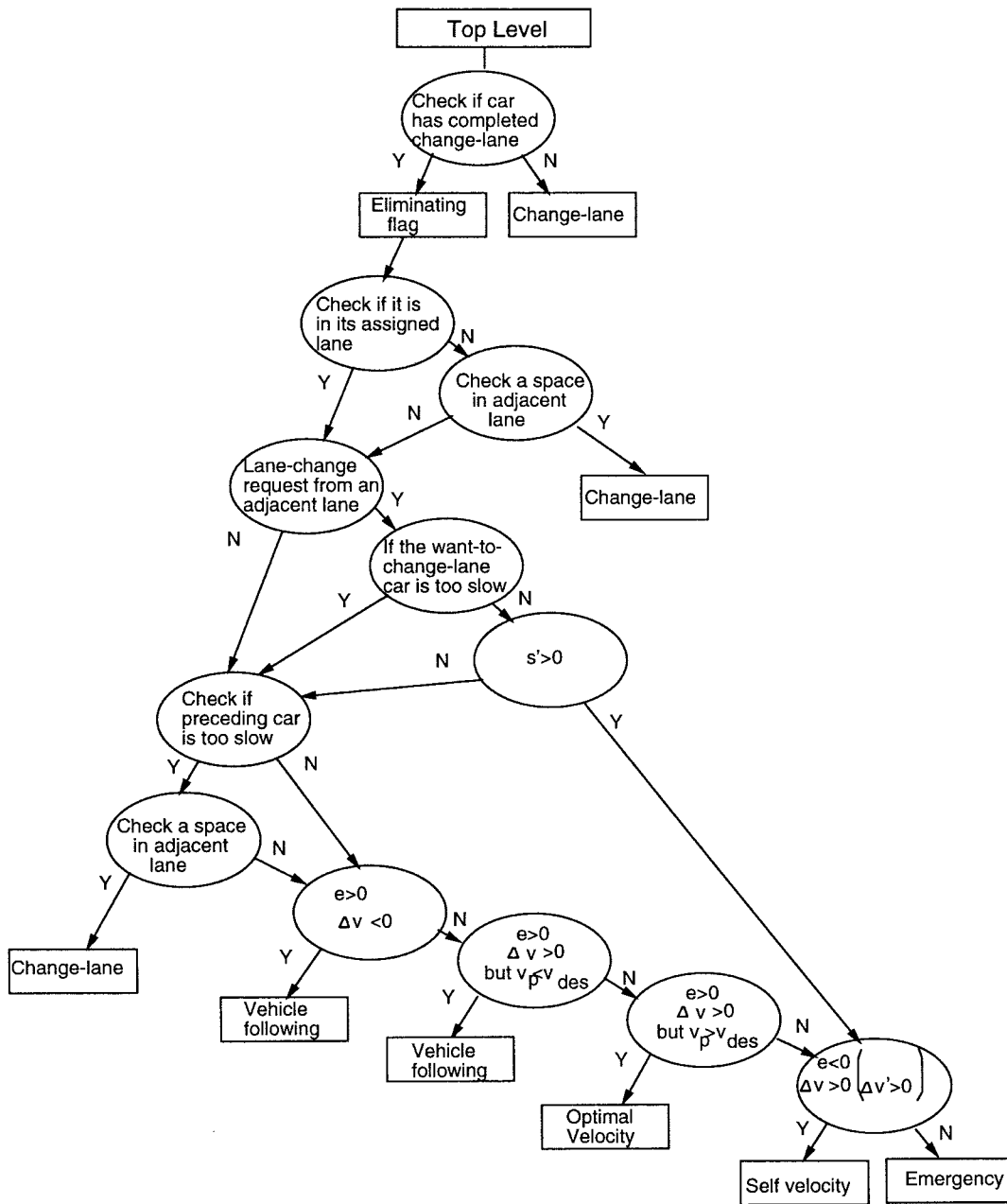


Fig. 2. Determining control mode.

- *Check for space in adjacent lane* is used to determine if there is a “safe” space (typically, 30 m) in an adjacent lane to accommodate merging.
- *Lane-change request from another vehicle* is used to check if the change-lane light is sent from another vehicle in an adjacent lane.
- *Check if preceding vehicle is too slow* is used to determine if the speed  $v_p$  of the preceding vehicle is far less than the desired speed  $v_{des}$ . We say that if  $v_p < v_{des} - 10$  m/s, then the immediate preceding vehicle is too slow.
- *If the want-to-change-lane car is too slow* is used to determine if the speed of a WANT-TO-CHANGE-LANE car in an adjacent lane is too slow. If *yes*, the controlled

car will not create a space for it. If *no*, the controlled vehicle will check if the longitudinal spacing between the controlled car and the WANT-TO-CHANGE-LANE car, denoted as  $s'$ , is greater than zero.

The other optional conditions are clear. By using these conditions, the goal of the decision is to infer the driving mode along the logic tree, as shown in Fig. 2. Let us explain five terminal objectives in the figure.

Define the following variables:

$$e = x_p - x_c - L_c - (\lambda_v v_c + H) \quad (3)$$

$$\Delta v = v_p - v_c \quad (4)$$

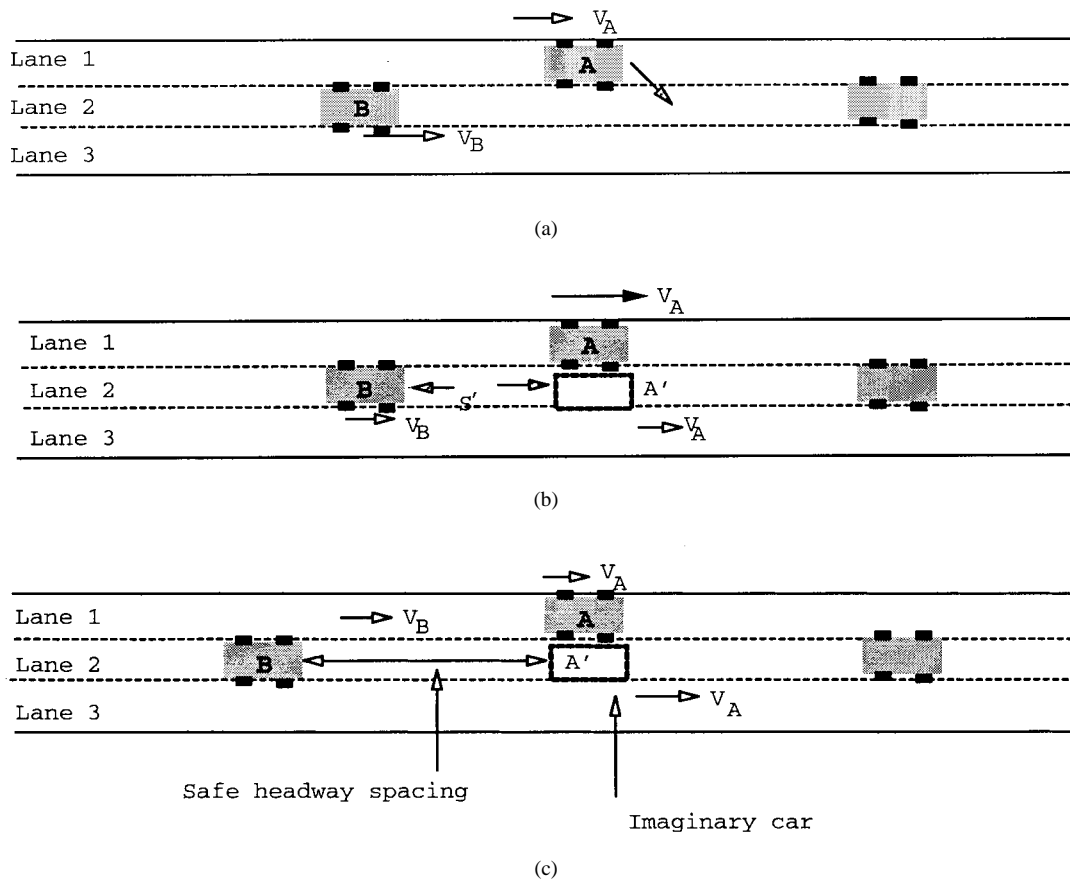


Fig. 3. Tracking an imaginary car in right lane.

where  $e$  is the error between the actual and safe distance,  $x_p(x_c)$  is the position of the preceding (current controlled) vehicle,  $L_c$  is the length of the controlled car,  $\lambda_v$  and  $H$  are positive constants,  $\Delta v$  is the relative velocity of the car in front, and  $v_p$  ( $v_c$ ) is the speed of the preceding (following) car.

The five control modes are the following.

- **Optimal Velocity:** This control mode includes two cases: 1) There is no car in the range of the distance sensor of the controlled car, and the car has to track the optimal velocity provided by user. In this case, we regard the traffic scenario as  $e > 0$  and  $\Delta v > 0$ . 2. The distance sensor might sense a preceding car, but  $e > 0$ . The velocity of the preceding car will possibly be faster than that of the speed of the controlled car. This corresponds to having  $e > 0$  and  $\Delta v > 0$ , but  $v_p > v_{des}$ .
- **Vehicle Following:** Suppose a car changes lane in front of it and make  $e > 0$  and  $\Delta v < 0$ , or  $e > 0$  and  $\Delta v > 0$ , but  $v_p < v_{des}$ . In this case, the control only needs a smooth action, since the controlled car is safe.
- **Self-Velocity:** Suppose there is no car in front of the controlled car for  $t < 0$ . At  $t = 0$ , a car changes lane in front of it at a distance which is less than the safe distance, but this car is moving faster than it. Thus, we maintain current speed of the controlled car so that safety and smooth control can be achieved at the same time. If

$s' > 0$  and  $\Delta v' > 0$ , where  $\Delta v'$  is the relative velocity of the car in an adjacent lane, then the controlled car also operates in this mode.

- **Emergency:** Suppose a car changes lane in front of the controlled car and is going slower, thereby making  $e < 0$  and  $\Delta v < 0$ . This scenario is critical from a safety point of view. Thus, we use a strong control action to maintain the safety. If  $s' > 0$  and  $\Delta v' < 0$ , then the guidance control also operates in this mode.
- **Change Lane:** We do not use a model to describe lateral behavior, since this paper focuses on longitudinal control. When a car receives a change-lane command coming from the mode selection, the simulation software will move the car into an adjacent lane while keeping its speed (we may see this scenario from Fig. 11).

Once the operating mode has been selected, the controller objective is to smoothly implement this mode without risking safety and passenger comfort. The details of designing controllers are given in the next section.

*Remark 1:* When a controlled car “sees” a WANT-TO-CHANGE-LANE light sent from another car in an adjacent lane, it will create a space to accommodate that cut-in car. In this case, the controlled car images a car ahead so as to implement the control objective using self-velocity (check if  $s' > 0$ ,  $\Delta v' > 0$ ) or emergency (check if  $s' > 0$ ,  $\Delta v' < 0$ ) control law. We demonstrate it using an example.

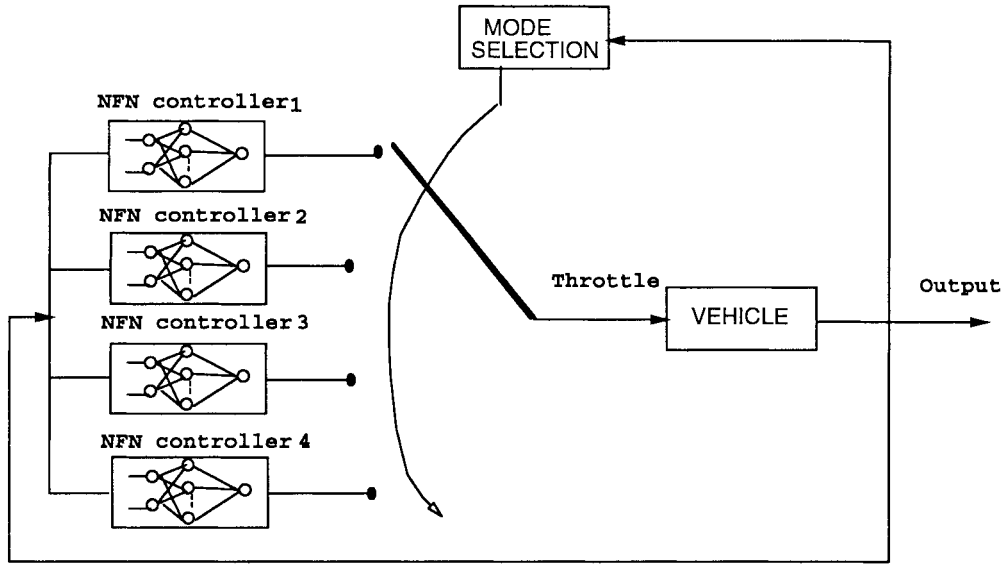


Fig. 4. Individual controllers for different modes.

Suppose  $A$  is in lane 1 and intends to change to lane 2 [see Fig. 3(a)]. It sends a WANT-TO-CHANGE-LANE light. When Car  $B$  detects this signal, it will stop tracking the car ahead in the same lane. Instead, Car  $B$  will determine how to create a space in lane 2 in order to accommodate  $A$ 's change of lane.  $B$ 's decision is to track an imaginary car ahead [see Fig. 3(b)]. The imaginary tracking objective is to achieve safe spacing [see Fig. 3(c)].

IV. CAR CONTROLLER

In this section, we concentrate on the car controller by designing individual controllers for longitudinal control that may be used for several modes of vehicle guidance operations. The NFN's are used to implement these controllers as shown in Fig. 4. Each controller design takes into account passenger and safety requirements. The goal of control is to take the system to the operating condition given by  $e = \Delta v = 0$ . We now describe the NFN constructs in detail.

A. Neural Fuzzy Learning Networks

In the introduction, we have discussed some models of neural fuzzy controllers that can be built from a set of input-output data pairs through parameter learning. We shall next focus mainly on the description of an NFN used in this paper.

Consider an NFN construct with fuzzy logic rules whose consequents are fuzzy singletons. Such fuzzy logic rules, called *fuzzy singleton rules*, are in the following form (see [17]):

$$R^j: \text{ IF } x_1 \text{ is } A_1^j \text{ AND } x_2 \text{ is } A_2^j \text{ AND } \dots \text{ AND } x_n \text{ is } A_n^j, \text{ THEN } y \text{ is } \bar{y}_j \tag{5}$$

where  $x_i$  is an input variable,  $y$  is the output variable,  $A_i^j$  are linguistic terms of the precondition part with membership

functions  $\mu_{A_i^j}(x_i)$ ,  $\bar{y}_j$  is a real number of the consequent part,  $j = 1, 2, \dots, M$ , and  $i = 1, 2, \dots, n$ . If product inference and a centroid defuzzifier are used, then the final output  $y^*$  of such a neural fuzzy system is calculated by

$$y^* = \frac{\sum_{j=1}^M \mu_j \bar{y}_j}{\sum_{j=1}^M \mu_j} \tag{6}$$

where

$$\mu_j = \mu_{A_1^j}(x_1) \mu_{A_2^j}(x_2) \dots \mu_{A_n^j}(x_n). \tag{7}$$

In order to obtain a standard output signal, the sigmoid function is used to force the NFN output to be within the range of (0, 1), that is

$$y = f(y^*) = \frac{1}{1 + \exp(-y^*)}. \tag{8}$$

The membership function  $\mu_{A_i^j}(\cdot)$  of the precondition part is expressed by Gaussian function

$$\mu_{A_i^j}(x_i) = \exp \left[ - \left( \frac{x_i - a_i^j}{b_i^j} \right)^2 \right] \tag{9}$$

where  $a_i^j$  and  $b_i^j$  are real-valued parameters.

The output of fuzzy reasoning  $y$  can be calculated by (6) and (7). Assume that  $y^d$  is the desired output for some input vector  $x = (x_1, x_2, \dots, x_n)^T$ . The objective function to be minimized is defined by

$$E = \frac{1}{2} (y - y^d)^2. \tag{10}$$

Hence, a similar learning algorithm to [17] can be derived using gradient rules:

$$\alpha_i^j(t+1) = \alpha_i^j(t) - \eta_a \frac{\partial E}{\partial \alpha_i^j} \quad (11)$$

$$b_i^j(t+1) = b_i^j(t) - \eta_b \frac{\partial E}{\partial b_i^j} \quad (12)$$

$$\bar{y}_j(t+1) = \bar{y}_j(t) - \eta_{\bar{y}} \frac{\partial E}{\partial \bar{y}_j} \quad (13)$$

where

$$\begin{aligned} \frac{\partial E}{\partial \alpha_i^j} &= \frac{\mu_j(x)}{M} (y - y^d) y (1 - y) (\bar{y}_j - y^*) \\ &\quad \sum_{j=1}^M \mu_j(x) \\ &\quad \cdot \frac{2(x_i - \alpha_i^j)}{b_i^{j3}} \end{aligned} \quad (14)$$

$$\begin{aligned} \frac{\partial E}{\partial b_i^j} &= \frac{\mu_j(x)}{M} (y - y^d) y (1 - y) (\bar{y}_j - y^*) \\ &\quad \sum_{j=1}^M \mu_j(x) \\ &\quad \cdot \frac{2(x_i - \alpha_i^j)^2}{b_i^{j3}} \end{aligned} \quad (15)$$

$$\frac{\partial E}{\partial \bar{y}_j} = \frac{\mu_j(x)}{M} (y - y^d) y (1 - y) \sum_{j=1}^M \mu_j(x) \quad (16)$$

The convergence speed of gradient algorithms is often slow. Several hours or even days of computer time are often required to train neural networks. In addition, the total number of iterations for learning an example in neural networks is often in the order of thousands. Therefore, it is an important problem to improve the learning performance of a neural network. In the next section, we will develop a more effective learning algorithm, that is, a mixed genetic/gradient algorithm.

### B. Mixed Genetic/Gradient Algorithm

GA's are global optimization techniques that avoid many shortcomings exhibited in conventional search techniques on a large and complex space. They simulate the procedures of nature evolution which ensure the proliferation of a quality solution via a systematic information exchange that depends on a probabilistic decision. In a simple GA, there are three most commonly used operators: reproduction, crossover, and mutation. The reproduction operator produces a new population from the old one by using the selection strategies which keep the worthy individuals in the next generation. The crossover operator provides a mechanism for strings to produce new genes through information exchange in a random way. The mutation operator chooses few members from the population pool according to the probability of mutation.

GA's are very robust due to the global searching. However, as it has been widely recognized, the GA involves intensive computation and is less efficient [24]. Improvement of the efficiency of the algorithm is very much needed [25]. A

solution to this problem is to integrate the GA into the gradient learning process. This approach can be addressed as follows.

#### Phase I

*Step 1:* An initial setting of fuzzy logic rules is done.

*Step 2—Chromosomal representation:* Each solution in the population is represented by a real number string rather than as a binary string. For  $x \in R^{1 \times n}$ ,  $\bar{y} \in R^{1 \times M}$ , the chromosomal representation may be expressed as an array

$$P = [w_1^T, \dots, w_n^T, w_{n+1}^T, \dots, w_{2n}^T, w_{2n+1}^T] \quad (17)$$

where  $w_i^T = (a_i^1, \dots, a_i^M)$ ,  $w_{n+i}^T = (b_i^1, \dots, b_i^M)$ ,  $i = 1, \dots, n$ ,  $w_{2n+1}^T = (\bar{y}_1, \dots, \bar{y}_M)$ . This kind of chromosomal representation has two advantages. One is that it guarantees that the domain expertise embodied in the representation will be preserved. The other is that the algorithm to be developed will feel natural to the designer.

*Step 3—Initial population:*  $N$  (an odd number) sets of parameter string  $P$  for the initial population are randomly generated.

*Step 4—Feedforward calculation of the NFN:* Fuzzy reasonings are performed for the input data  $(x_1, x_2, \dots, x_n)$  by using (6)–(8), where the membership value of  $\mu_i$  of each inference rule and the output of fuzzy reasoning  $y$  are then calculated.

*Step 5:* Calculate the fitness function for each chromosome. The fitness function ( $f(w_i)$ ) for the GA is defined as the average squared system error of the corresponding NFN

*Step 6—Selection:* According to the summation of total fitness, the parent  $P^l$  is selected from the population  $P$  in the  $t$ th generation. Any  $w_i'^T = w_i^T$  in  $P^l$  is chosen by a given random real number  $\alpha_i$  satisfying the following condition:

$$0 \leq \alpha_i \leq \sum_{j=1}^n f(w_j). \quad (18)$$

The index  $q$  is obtained from

$$q = \min \left\{ k | \forall k \in 1, \dots, n, \text{ s.t. } \alpha_i \leq \sum_{k=1}^n f(w_k) \right\}. \quad (19)$$

*Step 7—Crossover:* Perform the crossover using an average crossover function to produce the  $(N-1)/2$  offspring. The average crossover operator takes two parents which are selected in *Step 5* and produces one child that is the result of averaging the corresponding fields of two parents. In other words, the average crossover function is defined as

$$P_{cj} = \frac{P_{j+1} + P_j}{2}, \quad \text{for } j = 1, 2, \dots, \frac{N-1}{2}. \quad (20)$$

*Step 8—Mutation:* A real-number mutation operator, called a real-number creep, is used. The function we are optimizing is a continuous one with hills and valleys. If we are on a good hill, we want to jump around on it, to move nearer to the top. A real-number creep can have that effect. The mutation operation is defined as

$$P_{Mj} = P_{cj} + \rho_m \gamma_j, \quad \text{for } j = 1, 2, \dots, \frac{N-1}{2} \quad (21)$$

where  $\rho_m$  is the maximum value to be altered and  $\gamma_i \in [-1, 1]$  is a random variable with zero mean.

*Step 9—Termination checking:* Continue the cycle initiated in *Step 5* until convergence is achieved. The population is considered to have converged when the smallest value of the fitness functions in a population is less than the acceptable predefined value.

#### Phase II

*Step 1:* Initialize the weights using the chromosome with the smallest fitness function.

*Step 2—Training data:*  $(x_1, x_2, \dots, x_n, y^d)$  are input.

*Step 3—Feedforward calculation of the NFN:* Fuzzy reasoning is performed for the input data  $(x_1, x_2, \dots, x_n)$  by using (6)–(8). The membership value of  $\mu_i$  of each inference rule and the output of fuzzy reasoning  $y$  are then calculated.

*Step 4:* Tuning of the real number  $w_j$  of the consequent part is performed by using the gradient learning (16).

*Step 5:* The fuzzy reasoning conducted in *Step 3* is repeated.

*Step 6:* Tuning of the values  $a_i^j$  and  $b_i^j$  of the membership functions of the precondition part is done by substituting the tuned real number  $\bar{y}_j$  obtained in *Step 4*, the output  $y$ , the membership value  $\mu_i$ , and output data  $y^d$  into (14) and (15).

*Step 7:* The objective function (or the inference error)  $E(t) = 1/2[y(t) - y^d]^2$  is calculated, and *Steps 3–6* are repeated until its change  $\Delta E(t) = E(t) - E(t-1)$  is less than a desired threshold value.

In phase 1 of the hybrid learning algorithm, a GA is used to accelerate the whole learning process. The GA performs a global search and seeks a near-optimal initial point for the second stage. In this phase, each chromosome is used to construct the weights of the NFN. The fitness (objective) function for the GA is defined as the average squared system error of the corresponding neural network. Therefore, it becomes an optimization problem. The best fitness function value in a population is defined as the smallest value of the objective function in the current population.

After performing several iterations and meeting the stopping criteria, the first learning stage is terminated and the chromosome returning the minimum objective function (the best seed) is considered as the initial weights of the fuzzy neural network in the second stage. Next, the gradient algorithm (e.g., the backpropagation algorithm) performs the learning process until the terminal condition is satisfied.

#### C. Input–Output Structure and Design Considerations for the NFN Controllers

Since we use several individual controllers to achieve the vehicle guidance control, the corresponding controller structure should be discussed. In addition, it is very important to consider the ride quality for the passengers, while designing the vehicle controller. Failure to do so will result in an uncomfortable feeling for the passengers and even instability. The comfort constraints on the acceleration are determined by the potential of the car. For our design, we take them as  $[-5, 2]$   $\text{m/s}^2$ . However, the comfort constraints above may be violated due to a safety problem. In this case, the hardware constraints on the acceleration will replace the comfort constraints. We take them as  $[-10, 5]$   $\text{m/s}^2$ . With these in mind, we present

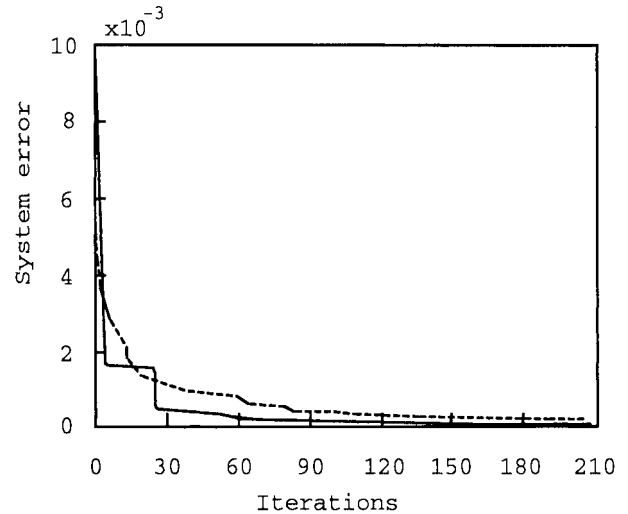


Fig. 5. System error for velocity controller design (solid line: hybrid learning; dashed line: gradient learning).

the input–output structure for each controller.

*1) Optimal Velocity Controller:* When a vehicle chooses traveling at the optimal speed, the control system uses the velocity and acceleration data measured by the on-board computer. The system output is the throttle force.

The two inputs for the velocity controller can be expressed as

$$\Delta v(t) = v_{\text{des}} - v_c(t) \quad (22)$$

$$a(t) = a_c(t) \quad (23)$$

where both  $\Delta v(t)$  and  $a(t)$  are the input,  $v_{\text{des}}$  is the desired optimal speed,  $v_c(t)$  is the controlled car's speed, and  $a_c(t)$  is the controlled car's acceleration. So, this controller defines the map  $Y: \mathbb{R}^2 \rightarrow \mathbb{R}$ . The acceleration should not exceed the comfort constraints.

*2) Tracking Self-Velocity Controller:* In this case, the car maintains its own speed by using the controller. The input–output is similar to the velocity controller.

The two inputs for this case can be expressed as

$$\Delta v(t) = v_c(t_0) - v_c(t) \quad (24)$$

$$a(t) = a_c(t) \quad (25)$$

where  $v_c(t_0)$  is the controlled car's speed at time  $t = t_0$ . The acceleration should not exceed the comfort constraints.

*3) Vehicle-Following Controller:* This controller maintains a headway distance between the controlled car and the preceding car. The controller uses the differences in acceleration and velocity between vehicles and the distance error as its inputs

$$c(t) = s_d - s_c(t), \quad s_d = \lambda_v v_c + H \quad (26)$$

$$\Delta v(t) = v_p(t) - v_c(t) \quad (27)$$

$$\Delta a(t) = a_p(t) - a_c(t) \quad (28)$$

where  $s_d$  is the desired headway spacing,  $s_c$  is the controlled car's spacing, and  $v_p(t)$  and  $a_p(t)$  are the speed and acceleration of the preceding car, respectively. So, this controller defines the map  $Y: \mathbb{R}^3 \rightarrow \mathbb{R}$ . Since one of the activating

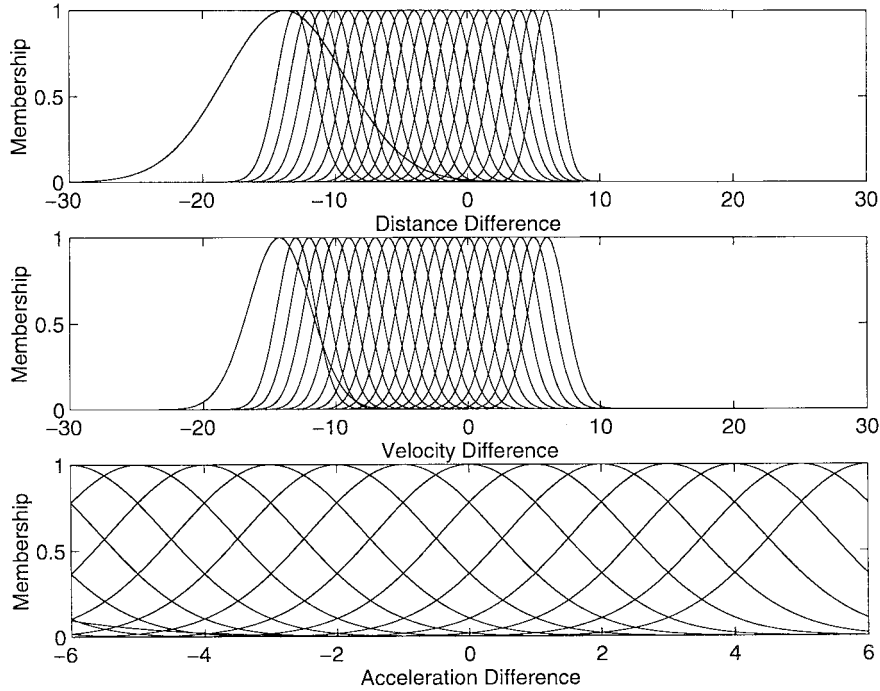


Fig. 6. Fuzzy sets for vehicle-following controller.

conditions for the vehicle following is  $e > 0$ , we use a weak control to achieve the safe distance. Thus, the acceleration should not exceed the comfort constraints.

4) *Emergency Controller*: This controller also maintains the same headway spacing as the above controller. The input–output structure is the same as the vehicle-following controller. However, the control action is strong, since the car is in an emergency scenario. The hardware constraints should be considered when designing the controller.

*Remark 2*: Since the structure and constraints of both the optimal velocity controller and self-velocity controller are the same, we merge them into a controller called the velocity controller. In addition, to transfer the range of (0, 1) into the range of throttle actuator  $(u_{\min}, u_{\max})$ , the control input is computed by

$$u = (u_{\max} - u_{\min})y + u_{\min}. \tag{29}$$

V. SIMULATION RESULTS

In this section, numerical simulations are carried out to verify the effectiveness of the NFN with mixed genetic/gradient parameter learning algorithm described previously. The parameters of vehicle dynamics used in the simulations are given in Section II. Before we begin the simulations, the NFN controller parameters must be specified by the algorithm stated previously.

A. *Parameter Learning for Velocity Controller*

The training data came from the car model as shown in Section II. The velocity controller gave 4000 training samples in 30 trajectories. We choose  $M = 11$  for the velocity

controller. The system error using the mixed genetic/gradient algorithm is shown in Fig. 5 (see solid line). The maximum mutation value of  $\rho_m$  is selected to be  $\rho_m = 0.11$ . After 27 iterations of the parameter learning process, the phase 1 of the mixed algorithm satisfied the terminating condition. The best fitness function was a value of about 0.068. Thus, its result was used as an initial weight vector in the second stage. After a total of about 200 iterations of the learning, the system error converges to a value  $2.7 \times 10^{-5}$ . The convergence speed of the gradient learning algorithm is also shown in Fig. 5 (see dashed line). Thus, the fuzzy set values for the fuzzy variables can be obtained.

B. *Parameter Learning for Vehicle-Following Controller*

A total of 5000 input–output sample pairs were used to train the NFN vehicle-following controller. The NFN system with  $M = 21$  has  $21 \times 4$  adjustable parameters. The maximum mutation value of  $\rho_m$  is selected to be  $\rho_m = 0.09$ . After 32 iterations of the parameter learning process, the first stage of the mixed algorithm satisfied the terminating condition. The best fitness function was a value of about 0.023. Thus, its result is used as an initial weight vector in the second stage. After a total of about 200 iterations of the learning, the system error converges to a value  $3.2 \times 10^{-5}$ . Fig. 6 depicts the membership functions created along the input variables of the NFN model.

C. *Parameter Learning for Emergency Controller*

Similar to the vehicle-following controller, the emergency controller has a strong control action with the hardware

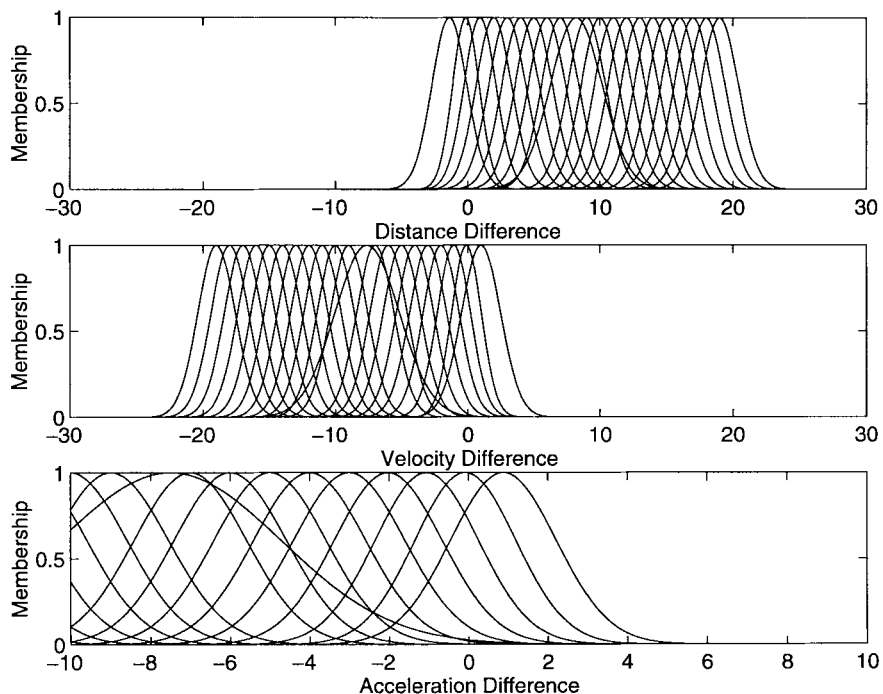


Fig. 7. Fuzzy sets for emergency controller.

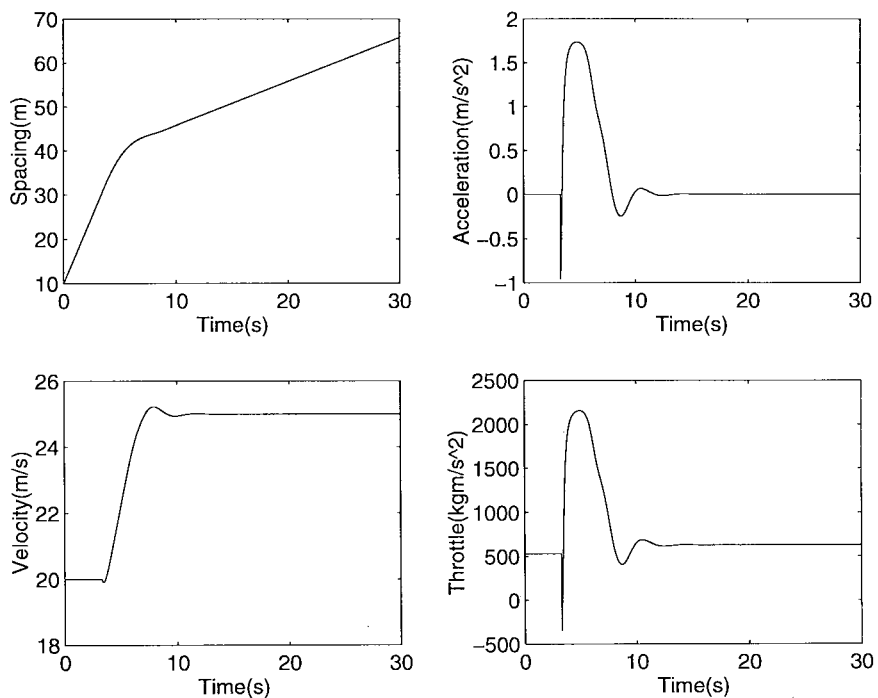


Fig. 8. Control performance of Example 1.

limitations. The same learning procedure as the above is carried out. Fig. 7 shows the system memberships for the NFN emergency controller.

*D. Vehicle Guidance Test for Car Controller*

Since an automatic vehicle must handle all possible traffic scenarios, we integrate the designed individual controllers into the vehicle guidance system and test the control performance

for different traffic scenarios. In the following examples, we define the vehicle equipped with the proposed NFN controllers as an NFN-car.

*Example 1:* The NFN-car is moving with a speed of 20 m/s. A car changes lanes in front of it, reducing the safety distance by 10 m, but it is going faster than the NFN-car by 6 m/s. Thus, the NFN-car uses the velocity controller (self-velocity mode) to keep its original velocity until the safety spacing (30

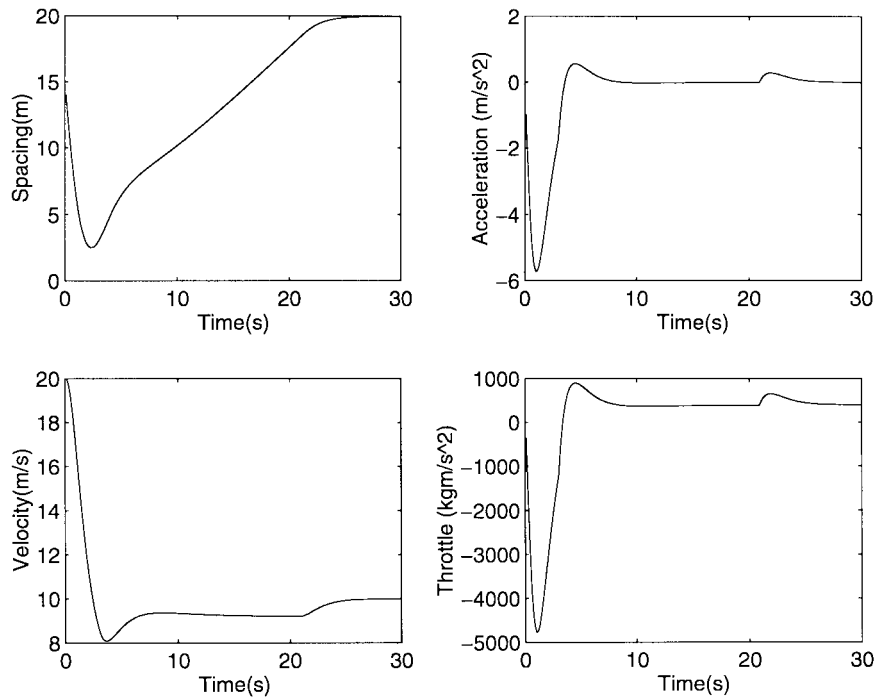


Fig. 9. Control performance of Example 2.

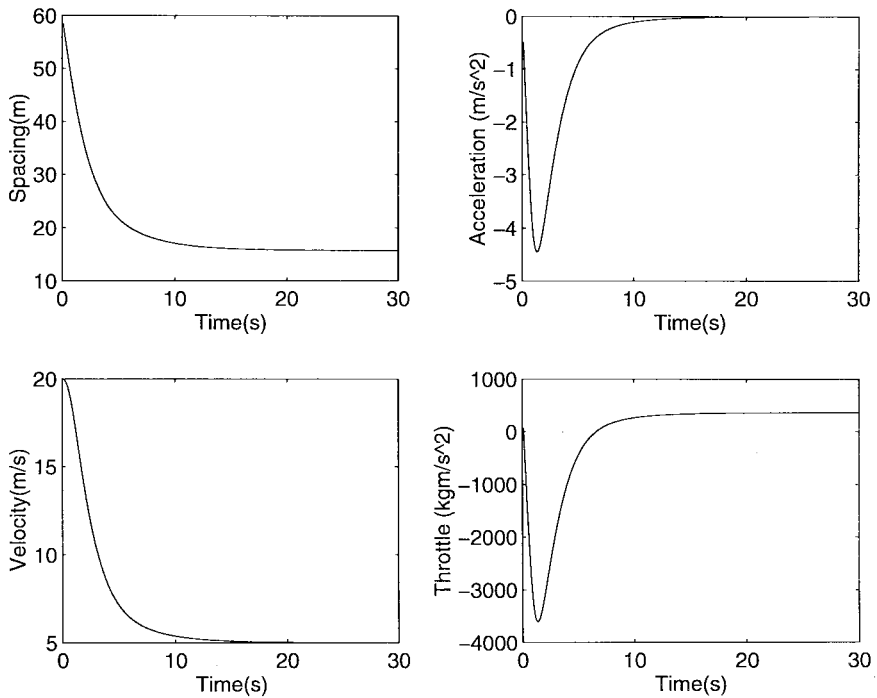


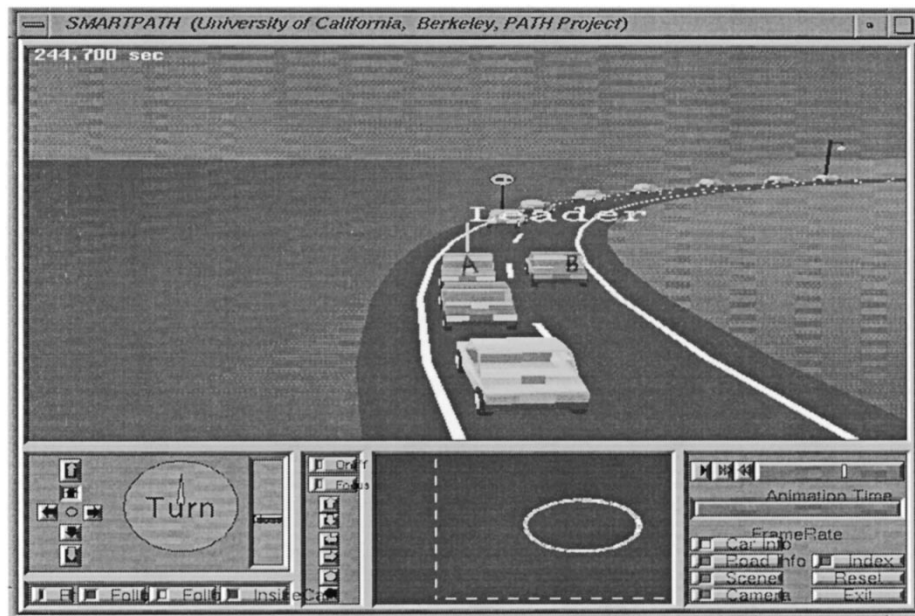
Fig. 10. Control performance of Example 3.

m) is satisfied. Then, it tracks the optimal velocity of 25 m/s using the velocity controller (optimal velocity mode). Fig. 8 shows the control result.

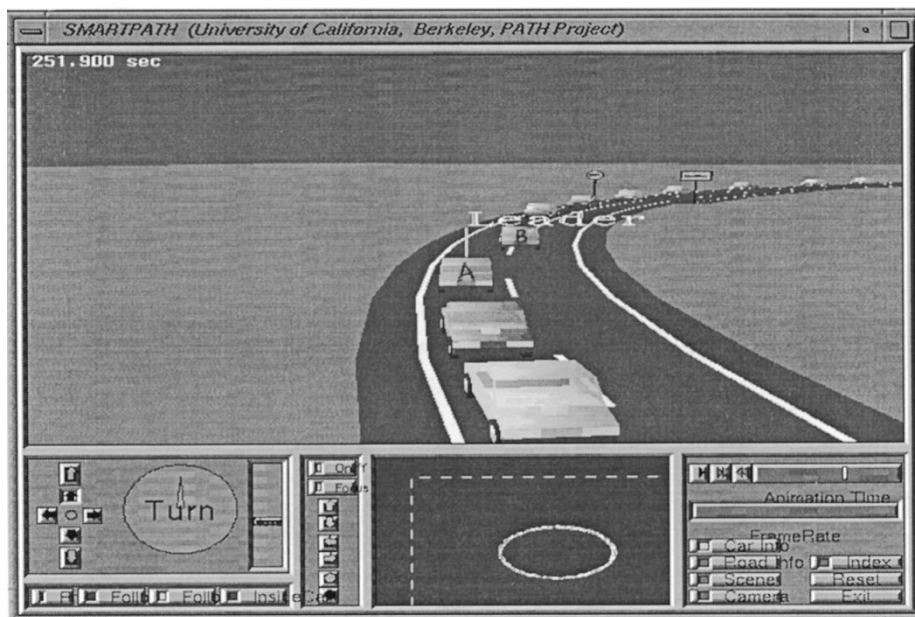
*Example 2:* The NFN-car is moving with at a speed of 20 m/s. A car changes lane in front of it at a distance which is 15 m less than the required safety distance (30 m) with the velocity going at a lower speed of 10 m/s. Initially, the

NFN-car operation is in the *emergency controller*, and then the *velocity controller* (operating in self-velocity mode) when  $\Delta v > 0$ . After the headway spacing is achieved, the NFN-car is operated by the *vehicle-following* controller. Fig. 9 shows the control result.

*Example 3:* The NFN-car is moving at a speed of 20 m/s. A car changes lane in front of it at a distance which is 60



(a)



(b)

Fig. 11. Visual simulation for complex traffic. (a) Car *B* sends a change lane light and Car *A* determines to create a space for Car *B*. (b) Car *B* finds enough space in left lane and then moves over.

m greater than the required safety distance (30 m) with the velocity of 5 m/s. The NFN-car operation is in the *vehicle-following* controller. Fig. 10 shows the control result.

Based on a simulation package, called SmartPath (see [27], [28]), the proposed guidance control system has been successfully implemented on a Silicon Graphics workstation. The program may be used to test the performance of an intelligent guidance car driving on a highway. Let us show a control result for complex traffic.

See Fig. 11(a). A guidance car defined as the leader, called Car *A*, detects a WANT-TO-CHANGE-LANE signal sent

from a car, called Car *B*, at time  $t = 244.7$  s. The speed of Car *B* is faster than that of Car *A*. Thus, Car *A* decides to create a space, as discussed in *Remark 1*, for Car *B*. The velocity controller is used so as to maintain Car *A*'s velocity. At time  $t = 251.9$  s, Car *B* finds that the space in its adjacent lane created by Car *A* is enough for moving over, and it determines to change lane. Thus, Car *A* starts to track Car *B* in its own lane [see Fig. 11(b)]. At time  $t = 269.0$ , the headway spacing between Cars *A* and *B* is achieved. Fig. 12 shows the control results of Car *A* and the velocity variation of Car *B*.

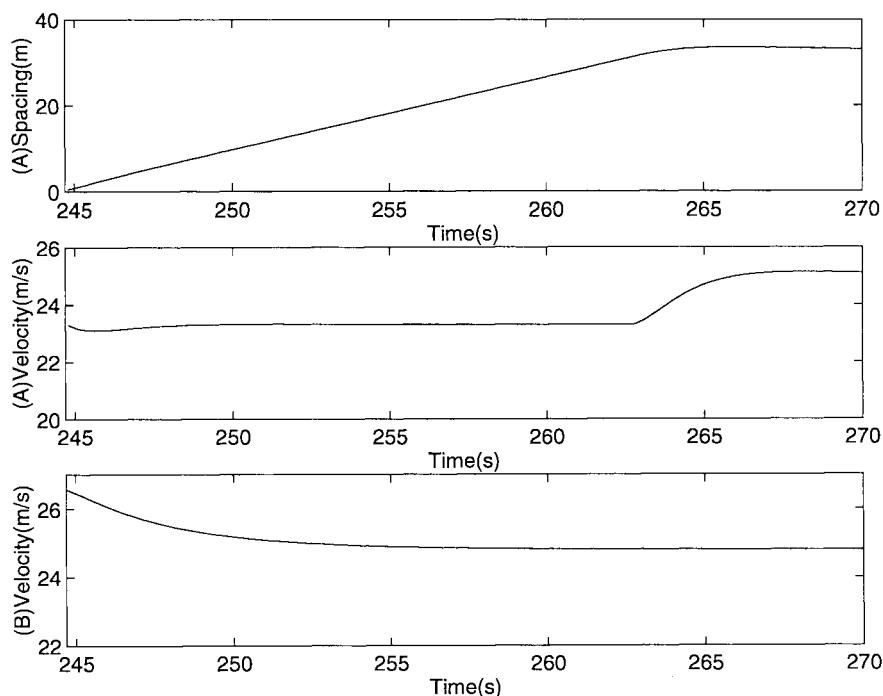


Fig. 12. Car A's control trajectories and Car B's velocity variation.

From Figs. 5–7, it is shown that the proposed learning scheme results in a more efficient training procedure. The main reason is that the GA can seek a good starting weight vector for the subsequent gradient learning. This improves the learning convergence speed as compared to the conventional pure learning (the backpropagation) algorithm. Furthermore, the hybrid learning can alter the membership functions to produce more efficient fuzzy control performance. The test results from Fig. 8–12 show that the designed controller can produce satisfactory tracking performance. It is also shown that the controller can guarantee traffic safety and produce smooth control behavior that results in a comfortable feeling for the passengers.

## VI. CONCLUSIONS

An integrating design of an intelligent vehicle control system has been described in this paper. It has been shown that both membership functions and control rule base in the NFN construct can be determined by a mixed genetic/gradient parameter algorithm, so as to obtain an optimal or near-optimal control performance in vehicle control. Based on this study, the design cycle time for the vehicle guidance system can be greatly reduced from hours to minutes. Computational simulations have shown the effectiveness of the designed controller. An extension of this study to a more complex vehicle driving system that can handle information exchanged with a traffic management center will be investigated in future work.

## ACKNOWLEDGMENT

The authors wish to take this opportunity to thank Prof. O. Kaynak for his helpful suggestions.

## REFERENCES

- [1] P. Varaiya and S. E. Shladover, "Sketch of an IVHS systems architecture," PATH, Berkeley, CA, Res. Rep. UCB-ITS-PRR-91-3, 1991.
- [2] P. Varaiya, "Smart cars on smart roads: problems for control," *IEEE Trans. Automat. Contr.*, vol. 38, pp. 195–207, Feb. 1993.
- [3] S. N. Huang and W. Ren, "Design of vehicle following control systems with actuator delays," *Int. J. Syst. Sci.*, vol. 28, no. 2, pp. 145–151, 1997.
- [4] P. Ioannou, Z. Xu *et al.*, "Intelligent cruise control: Theory and experiment," in *Proc. 32nd IEEE Conf. Decision and Control*, San Antonio, TX, 1993, pp. 1885–1890.
- [5] P. Ioannou and Z. Xu, "Adaptive throttle control for automatic vehicle following," in *Proc. 12th Triennial World Congr. Int. Federation of Automatic Control*, Sydney, Australia, 1994, pp. 113–116.
- [6] W. Ren and D. Green, "Continuous platooning: A new evolutionary and operating concept for automated highway systems," Dep. Elect. Eng. Comput. Sci., Univ. California, Berkeley, Memo. UCB/ERL M94/24, 1994.
- [7] J. Forbes, T. Huang, K. Kanazawa, and S. Russell, "The BATmobile: Toward a bayesian automated taxi," in *Proc. 14th Int. Joint Conf. Artificial Intelligence*, Montreal, P. Q., Canada, 1995, pp. 418–423.
- [8] A. Niehaus and R. F. Stengel, "Probability-based decision making for automated highway driving," *IEEE Trans. Veh. Technol.*, vol. 43, pp. 626–634, Aug. 1994.
- [9] ———, "An expert system for automated highway driving," *IEEE Contr. Syst. Mag.*, vol. 11, pp. 53–60, Apr. 1991.
- [10] F. Eckhard and M. Robert, "Nonlinear path control in automated vehicle guidance," *IEEE Trans. Robot. Automat.*, vol. 13, pp. 49–60, Feb. 1997.
- [11] S. N. Huang and W. Ren, "Safety, comfort, and optimal tracking control in AHS applications," *IEEE Contr. Syst. Mag.*, vol. 18, pp. 50–64, Aug. 1998.
- [12] S. Kikuchi and P. Chakroborty, "Car-following model based on fuzzy inference system," National Research Council, Washington, DC, Transportation Research Record 1365, TRB, 1994, pp. 82–91.
- [13] H. M. Kim, J. Dickerson, and B. Kosko, "Fuzzy throttle and brake control for platoons of smart cars," *Fuzzy Sets Syst.*, vol. 84, no. 23, pp. 209–234, Dec. 1996.
- [14] J. S. R. Jang, "ANFLS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, pp. 665–685, May/June 1993.
- [15] ———, "Self learning fuzzy controller based on temporal back propagation," *IEEE Trans. Neural Networks*, vol. 3, pp. 156–159, Jan. 1993.

- [16] J. S. R. Jang, C. T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Upper Saddle River, NJ: Prentice-Hall, 1997.
- [17] L. X. Wang, *Adaptive Fuzzy Systems and Control: Design and Stability Analysis*. Englewood Cliffs, NJ: PTR Prentice-Hall, 1994.
- [18] C. T. Lin and C. S. G. Lee, "Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems," *IEEE Trans. Fuzzy Syst.*, vol. 2, pp. 46–63, Jan. 1995.
- [19] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
- [20] D. Park, A. Kandel, and G. Langholz, "Genetic-based new fuzzy reasoning models with application to fuzzy control," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, pp. 39–47, Jan. 1994.
- [21] K. S. Tang, K. F. Man, and C. Y. Chan, "Fuzzy control of water pressure using genetic algorithm," in *Proc. IFAC Workshop Safety, Reliability, and Applications of Emerging Intelligent Control Technologies*, Hong Kong, Dec. 1994, pp. 15–20.
- [22] C. K. Chiang, H. Y. Chung, and J. J. Lin, "Self-learning fuzzy logic controller using genetic algorithms with reinforcements," *IEEE Trans. Fuzzy Syst.*, vol. 5, pp. 460–467, Aug. 1997.
- [23] S. B. Choi and J. K. Hedrick, "Robust throttle control of automotive engines: Theory and experiments," *ASME J. Dynam. Syst., Meas., Contr.*, vol. 118, no. 1, pp. 92–98, 1996.
- [24] S. W. Wilson, "Quasi-Darwinian learning in a classifier system," in *Proc. 4th Int. Workshop Machine Learning*, 1987, pp. 59–65.
- [25] A. D. McAulay and J. C. Oh, "Improving learning of genetic rule-based classifier systems," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, pp. 152–159, Jan. 1994.
- [26] S. Shiehholeslam, "Control of a class of interconnected nonlinear dynamical systems: the platoon problem," Ph.D. dissertation, Dep. Elect. Eng. Comput. Sci., Univ. California, Berkeley, 1991.
- [27] F. Eskafi, D. Khorramabadi, and P. Varaiya, "Smartpath: An automated highway system simulator," Dep. Elect. Eng. Comput. Sci., Univ. California, Berkeley, Tech. Note 94-3, 1994.
- [28] F. Eskafi and D. Khorramabadi, *Smartpath User's Manual*, Dep. Elect. Eng. Comput. Sci., Univ. California, Berkeley, and PATH/ITS, UCB-94, 1994.



**Sunan Huang** received the Ph.D. degree from Shanghai Jiao Tong University, Shanghai, China, in 1994.

He was a Postdoctoral Fellow at Zhejiang University from 1994 to 1995 and a Visiting Postdoctoral Fellow at the University of California, Berkeley, from 1995 to 1997. He is currently a Postdoctoral Fellow at the National University of Singapore, Singapore. His research interests include automated vehicle control and process control.



**Wei Ren** (SM'97) received the Ph.D. degree from the University of Illinois, Urbana-Champaign, in 1991.

Since 1991, he has been an Assistant Professor in the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley. His research interests include statistical adaptive control, signal processing problems in audio systems, and transportation systems.

Prof. Ren was the recipient of a Knowles Fellowship in Acoustics and the Robert Chien Memorial Award for excellence in doctoral research from the University of Illinois. He received the Best Student Paper Award at the 1991 Conference on Decision and Control. He is also the recipient of an NSF Research Initiation Award and an AT&T Faculty Award.