

Fuzzy throttle and brake control for platoons of smart cars

Hyun Mun Kim, Julie Dickerson¹, Bart Kosko*

*Signal and Image Processing Institute, Department of Electrical Engineering-Systems, University of Southern California,
Los Angeles, CA 90089-2564, USA*

Received May 1995; revised August 1995

Abstract

Additive fuzzy systems can control the velocity and the gap between cars in single-lane platoons. The overall system consists of throttle and brake controllers. We first designed and tested a throttle-only fuzzy system on a validated car model and then with a real car on highway I-15 in California. We used this controller to drive the “smart” car on the highway in a two-car platoon. Then we designed a throttle and brake controller. The combined system controls the platoon on downhill parts of the freeway and as it decelerates to slower speeds. We modeled the brake controller using the real test data from the brake system. A logic switch for throttle and brake decides which system to use. The gap controller uses data only from its own sensors and there is no communication among cars. The simulation results show that follower cars with a combined brake/throttle controller can maintain a constant gap when the platoon goes downhill and slows. An adaptive throttle controller uses neural systems to learn the fuzzy rules for different vehicle types.

Keywords: Adaptive fuzzy systems; Smart-car platoons; Model-free control; Ellipsoidal rules; Unsupervised clustering; Supervised gradient descent; Standard additive model; Function approximation

1. Introduction

Traffic clogs highways around the world. Platoons of cars can increase the flow and mean speed on freeways. A platoon is a group of cars with a lead car and one or more follower cars that travel in the same lane. Electronic links tie the cars together. Computer control speeds their response times to road hazards so that the cars can travel more safely on their own or in platoon groups. The lead car plans the course for the platoon. It picks

the velocity and car spacing and picks which maneuvers to perform. Platoons use four maneuvers: merge, split, velocity change, and lane change [10, 11]. A merge combines two platoons into one. A split splits one platoon into two. A lane change moves a single car into an adjacent lane. Combined maneuvers help cars move through traffic.

Standard control systems use an input–output math model of the car “plant” and its environment. Fuzzy systems do not use an input–output math model or exact car parameters. A fuzzy system $F: R^n \rightarrow R^p$ is a set of fuzzy rules that maps inputs to outputs [14]. Fuzzy systems give a model-free estimate of a nonlinear control function. They compute a conditional mean as Appendix A

* Corresponding author.

¹ Julie Dickerson is with the Electrical and Computer Engineering Department, Iowa State University, Ames, IA 50011.

shows: $F(x) = E[Y|X = x]$. The fuzzy platoon controller uses rules that act like the skills of a human driver. The rules have the form “If input conditions hold to some degree then output conditions hold to some degree” or “If X is A then Y is B ” for multivalued or “fuzzy” sets A and B . Each fuzzy rule defines a fuzzy patch or a Cartesian product $A \times B$ in the input–output state space $X \times Y: A \times B \subset X \times Y$. To approximate a function f the fuzzy system F covers the graph of f with fuzzy patches and averages patches that overlap [15].

The fuzzy platoon controller drives a car in or out of the platoon and acts as a distributed control system for future freeways. It includes an integrated maneuver controller for course selection and an individual vehicle controller for throttle, brake, and steering control as shown in Fig. 1. We implemented the individual vehicle controller only.

We designed a fuzzy controller for gap control using throttle only. The gap controller gets data from its own sensors. We tested the fuzzy gap controller on Interstate-15 in Escondido, California. The controlled car followed the lead car as it changed speed and went over hills. The system performed smoothly in all cases. But when it went downhill the controlled car got close to the leader.

We next designed a throttle and brake controller. The combined system lets us control platoons on downhill parts of the freeway and during decelerations to slower speeds. We simulated the brake controller using the real test data from the brake

system. A logic switch for throttle and brake picks when to use each system. This logic switch avoids frequent oscillations between the throttle and the brake controller. Simulation results showed that the follower car using throttle only could not slow down enough to avoid hitting the leader car on the downhill grade. The follower car with a combined brake/throttle controller did not collide with the lead car.

We hand tuned the fuzzy sets and rules for the gap controller. Neural learning can also find the rules from system input–output data. We used a neural-fuzzy system that tuned the fuzzy rules for the velocity controller with unsupervised and supervised learning. A hybrid system [4] used unsupervised learning to quickly pick the first set of ellipsoidal fuzzy rules. Then supervised learning tuned the rules using gradient descent. Each rule defines a fuzzy subset or connected region of state space and thus relates throttle response, acceleration, and velocity. Section 3 describes the hybrid ellipsoidal learning system. The appendices formally derive the general fuzzy system and its learning laws.

2. Additive fuzzy systems

A scalar-valued fuzzy system $F: R^n \rightarrow R$ stores m rules of the word from “If $X = A_j$ then $Y = B_j$ ” or the patch form $A_j \times B_j \subset X \times Y = R^n \times R$. The if-part fuzzy sets $A_j \subset R^n$ and then-part fuzzy sets $B_j \subset R$ have arbitrary set functions $a_j: R^n \rightarrow [0, 1]$ and $b_j: R \rightarrow [0, 1]$. The system can use the joint set function a_j or some factored form such as $a_j(x) = a_j^1(x_1) \cdots a_j^n(x_n)$ or $a_j(x) = \min(a_j^1(x_1), \dots, a_j^n(x_n))$ or any other conjunctive form for input vector $x = (x_1, \dots, x_n) \in R^n$. Our fuzzy systems (like most) use min to form a_j from the coordinate set functions a_j^i . Product tends to work better for exponential or Gaussian set functions.

An additive fuzzy system [14, 15] sums the unweighted “fired” then-part sets B_j :

$$B = \sum_{j=1}^m B_j = \sum_{j=1}^m a_j(x) B_j. \tag{1}$$

Fig. 2 shows the parallel fire-and-sum structure of the SAM system. A vector input x matches or

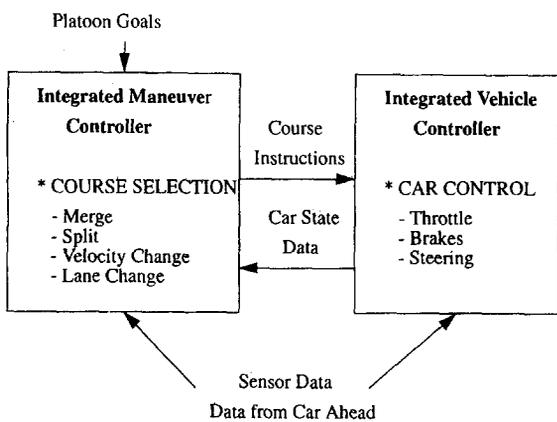


Fig. 1. Block diagram of the fuzzy platoon controller.

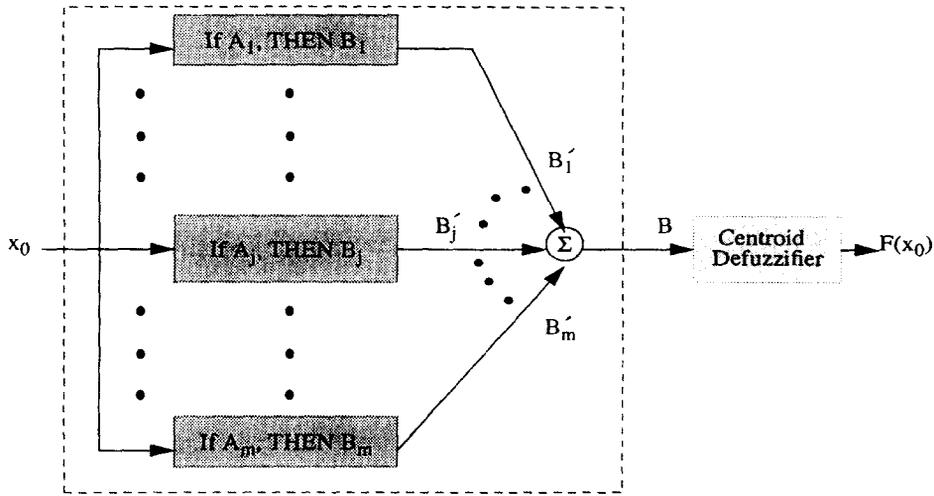


Fig. 2. Architecture of an additive fuzzy system $F: R^n \rightarrow R^p$ with m rules. Each input $x_0 \in R^n$ enters the system F as a numerical vector. At the set level x_0 acts as a delta pulse $\delta(x - x_0)$ that combs the if-part fuzzy sets A_j and gives the m set values $a_j(x_0)$: $a_j(x_0) = \int_{R^n} \delta(x - x_0) a_j(x) dx$. The set values “fire” the then-part fuzzy sets B_j to give B'_j . A standard additive model (SAM) scales each B_j with $a_j(x)$. The system then sums the B'_j sets to give B . The system output $F(x_0)$ is the centroid of B .

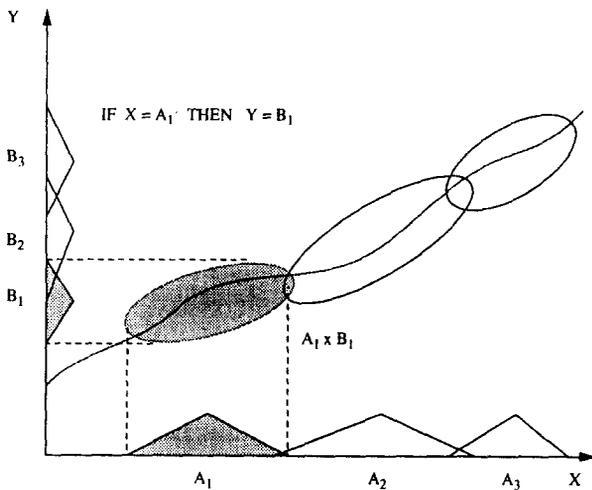


Fig. 3. Each fuzzy rule defines a Cartesian-product patch or fuzzy subset of the input–output state space. The fuzzy system approximates a function as it covers its graph with rule patches. Lone optimal rule patches cover extrema.

“fires” the if-part sets A_j of each rule. The system sums the scaled then-part sets B'_j and takes the centroid of the summed sets to give the output $F(x)$. These systems can uniformly approximate any continuous (or bounded measurable) function f on a compact domain [15].

Fig. 3 shows how three rule patches can cover part of the graph of a scalar function $f: R \rightarrow R$. The patch cover shows that all fuzzy systems $F: R^n \rightarrow R^p$ suffer from *rule explosion* in high dimensions. A fuzzy system F needs on the order of k^{n+p-1} rules to cover the graph and thus to approximate a vector function $f: R^n \rightarrow R^p$ where k is the number of sets in each dimension. Optimal rules can help deal with the exponential rule explosion. Lone or local mean-squared optimal rule patches cover the extrema [16] of the approximated f — they “patch the bumps”. Better learning schemes move rule patches to or near extrema and then fill in between extrema with extra rule patches as the rule budget allows.

The scaling choice $B'_j = a_j(x)B_j$ gives a *standard additive model* or SAM. Appendix A shows that taking the centroid of B in (1) gives [14–17] the SAM ratio

$$F(x) = \frac{\sum_{j=1}^m a_j(x) V_j c_j}{\sum_{j=1}^m a_j(x) V_j} \tag{2}$$

V_j is the nonzero volume or area of the then-part set B_j . c_j is the centroid of B_j or its center of mass. The ratio (2) reduces to the “center of gravity” model of Sugeno [21] and others if $V_1 = \dots = V_m > 0$.

The SAM theorem (2) implies that the fuzzy structure of the then-part sets B_j does not matter. The ratio depends on just the volume V_j and location c_j of the then-part sets B_j . We need to pick the scalar centers c_j and the volumes V_j . Appendix B uses gradient descent to derive the supervised learning laws that tune the SAM parameters a_j , V_j , and c_j . The next section shows how to apply these general learning schemes for egg-shaped or ellipsoidal fuzzy rules.

3. Learning fuzzy rules

Fuzzy rules can come from brains or brain-like systems. We can ask an expert for the if-then rules or we can act as the experts ourselves and try to state the rules and tune them [9]. Or we can use a neural (statistical) system to learn the rules from training data. This section describes a neural-fuzzy system that learns fuzzy rules with both unsupervised and supervised learning shown in Fig. 4.

3.1. Unsupervised rule estimation with competitive learning

A fuzzy rule patch can take the form of an ellipsoid [4]. This trades the generality of fuzzy rule patches for the mathematical simplicity of quadratic forms. A positive definite matrix P defines an ellipsoid in the q -dimensional input–output state space where $q = n + p$ (Fig. 5). The ellipsoid is the locus of all z that satisfy [2]

$$\alpha^2 = (z - m)^T P (z - m), \tag{3}$$

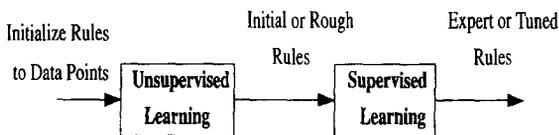


Fig. 4. A neural-fuzzy system can learn and tune the fuzzy rules with a hybrid of unsupervised and supervised learning. Unsupervised competitive learning initializes the supervised gradient descent search for rules that locally minimize the mean-squared error of the function approximation.

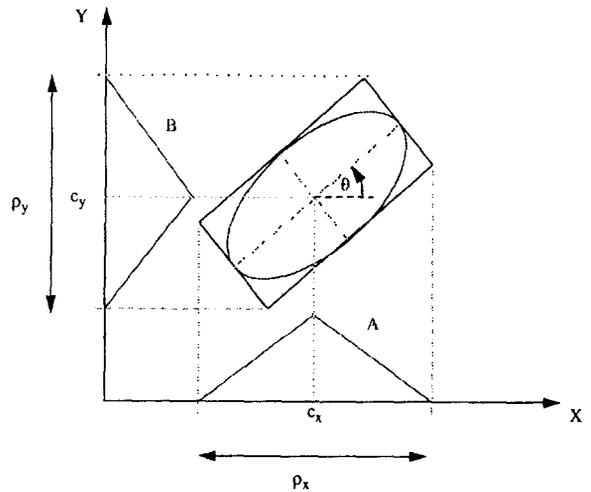


Fig. 5. A positive definite matrix defines an ellipsoid about the center m of the ellipsoid. The eigenvalues of the matrix define the length of the axes. The projections of the ellipsoid onto the axes define the input and output fuzzy sets A and B . ρ_x and ρ_y are the lengths of the ellipsoid's projections onto the x and y axes.

where α is a positive real number and m is the center of the ellipsoid in R^q . P has eigenvalues $\lambda_1, \dots, \lambda_q$. The eigenvalues define the ellipsoid axes. The Euclidean half-lengths of the axes equal $\alpha/\sqrt{\lambda_1}, \dots, \alpha/\sqrt{\lambda_q}$. To simplify the math we used a hyperrectangle to circumscribe the ellipsoid. The projections onto the input axes form the fuzzy sets. We used symmetric triangular sets centered at m to approximate these ellipsoidal “shadows”. The unit eigenvectors define direction cosines for each axis of the ellipse. The direction cosine $\cos \gamma_{kij}$ is the angle between the j th eigenvector and the i th axis for the k th ellipsoid. The projection of the k th hyperrectangle onto the i th axis has length ρ_{ki} :

$$\rho_{ki} = 2\alpha \sum_{j=1}^q \frac{|\cos \gamma_{kij}|}{\sqrt{\lambda_{kj}}}. \tag{4}$$

Adaptive vector quantization (AVQ) systems adaptively cluster pattern data in a state space. An autoassociative AVQ system combines the input x and the output y of the data to form $z^T = [x^T \ y^T]$. Competitive learning estimates the first- and second-order statistics of the data with the stochastic difference equations for the winning

neuron [14]

$$q_j(t + 1) = q_j(t) + \mu_t [z(t) - q_j(t)], \quad (5)$$

$$\begin{aligned} K_j(t + 1) \\ = K_j(t) + v_t [(z(t) - q_j(t))(z(t) - q_j(t))^T - K_j(t)]. \end{aligned} \quad (6)$$

The coefficients μ_t and v_t must satisfy the convergence conditions of stochastic approximation [14]. In practice $\mu_t \approx 1/t$ and $v_t \approx 1/t$. Appendix B gives the details of this learning method.

3.2. Supervised ellipsoidal learning

A supervised neural system learns the ellipsoidal rules as it locally minimizes the mean-squared error of the function approximation. The neural system learns the size and shape of the fuzzy rule patches that minimize the error. The gradient descent algorithm [14] takes the gradient of the instantaneous mean-squared error SE_k :

$$SE_k = \frac{1}{2} (d_k - F(x_k))^2. \quad (7)$$

Here d_k is the desired output of the system. $F(x_k)$ is the output of the additive fuzzy system with input x_k . Gradient descent estimates the eigenvalues, rotation angles, and centroids of the ellipsoidal patches [4].

We assume that the fuzzy sets are the triangular projections of the bounding hyperrectangles around the ellipsoids on the axes of the state space. The volume of the i th triangular output set is V_i :

$$V_i = \frac{1}{2} \rho_{iq} = \alpha \sum_{j=1}^q \frac{|\cos \gamma_{kij}|}{\sqrt{\lambda_{kj}}}. \quad (8)$$

ρ_{iq} is the base of the i th fuzzy rule patch projection on the q th or output axis in (4) and γ_{kij} are the direction cosines. The *fit* (fuzzy unit) value $a_j(x)$ is the degree to which input x belongs to the i th fuzzy set:

$$a_j(x) = \min_i (a_j^i(x)). \quad (9)$$

a_j^i is the triangular input fuzzy set for the i th ellipsoid projected on the j th axis:

$$a_j^i(x) = \begin{cases} 1 - \frac{|x - c_{x_{ij}}|}{\rho_{ij}} & \text{for } |x - c_{x_{ij}}| \leq \rho_{ij}/2, \\ 0 & \text{else.} \end{cases} \quad (10)$$

The supervised algorithm uses an iterative form of gradient descent:

$$E_i(k + 1) = E_i(k) + A_{c_k} \Delta_{E_i} SE_k. \quad (11)$$

$E_i(k)$ is a concatenated vector of the i th ellipsoid's parameters. A_{c_k} is a diagonal matrix of decreasing learning coefficients. $E_i(k)$ contains the eigenvalues, the centroid vector, and the independent orientation angles of the i th ellipsoid.

The chain rule of differential calculus gives the supervised ellipsoidal algorithm

$$\delta \lambda_{ij}^k(k) = -\frac{\partial SE_k}{\partial \lambda_{ij}^k} = -\frac{\partial SE_k}{\partial F_k} \left[\frac{\partial F_k}{\partial a_i^k} \frac{\partial a_i^k}{\partial \lambda_{ij}^k} + \frac{\partial F_k}{\partial V_i^k} \frac{\partial V_i^k}{\partial \lambda_{ij}^k} \right], \quad (12)$$

$$\delta c_{x_i}^k(k) = -\frac{\partial SE_k}{\partial c_{x_i}^k} = -\frac{\partial SE_k}{\partial F_k} \frac{\partial F_k}{\partial a_i^k} \frac{\partial a_i^k}{\partial c_{x_i}^k}, \quad (13)$$

$$\delta c_{y_i}^k(k) = -\frac{\partial SE_k}{\partial c_{y_i}^k} = (d_k - F_k) \frac{V_i^k a_i^k}{\sum_{j=1}^r V_j^k a_j^k}, \quad (14)$$

$$\delta \gamma_{ijl}^k(k) = -\frac{\partial SE_k}{\partial \gamma_{ijl}^k} = -\frac{\partial SE_k}{\partial F_k} \left[\frac{\partial F_k}{\partial a_i^k} \frac{\partial a_i^k}{\partial \gamma_{ijl}^k} + \frac{\partial F_k}{\partial V_i^k} \frac{\partial V_i^k}{\partial \gamma_{ijl}^k} \right]. \quad (15)$$

The partial derivatives [4] in (12)–(15) have the SAM form

$$\frac{\partial SE_k}{\partial F_k} = -(d_k - F_k), \quad (16)$$

$$\frac{\partial F_k}{\partial a_i^k} = \frac{V_i^k \sum_{j=1}^r V_j^k a_j^k (c_{y_i}^k - c_{y_j}^k)}{(\sum_{j=1}^r V_j^k a_j^k)^2}, \quad (17)$$

$$\frac{\partial F_k}{\partial V_i^k} = \frac{a_i^k \sum_{j=1}^r V_j^k a_j^k (c_{y_i}^k - c_{y_j}^k)}{(\sum_{j=1}^r V_j^k a_j^k)^2}, \quad (18)$$

$$\frac{\partial a_i^k}{\partial \lambda_{ij}^k} = \begin{cases} -\frac{2\alpha_i |x - c_{x_{ii}}| |\cos \gamma_{ij}|}{\rho_{ii}^2 \sqrt{\lambda_{ij}^3}} & \text{if } |x - c_{x_{ii}}| \leq \rho_{ii}/2, \\ 0 & \text{if } |x - c_{x_{ii}}| > \rho_{ii}/2, \end{cases} \quad (19)$$

$$\frac{\partial a_i^k}{\partial c_{x_{ii}}^k} = \begin{cases} \frac{2}{\rho_{ii}} & \text{if } 0 < x - c_{x_{ii}} \leq \rho_{ii}/2, \\ 0 & \text{else,} \\ -\frac{2}{\rho_{ii}} & \text{if } -\rho_{ii}/2 < x - c_{x_{ii}} \leq 0, \end{cases} \quad (20)$$

$$\frac{\partial V_i^k}{\partial \gamma_{ij}^k} = \begin{cases} -\frac{\alpha_i \sin \gamma_{ij}}{\sqrt{\lambda_{ij}}} & \text{if } |\gamma_{ij}| \leq \pi/2, \\ \frac{\alpha_i \sin \gamma_{ij}}{\sqrt{\lambda_{ij}}} & \text{if } |\gamma_{ij}| > \pi/2, \end{cases} \quad (21)$$

$$\frac{\partial m_i^k}{\partial \gamma_{ij}^k} = \begin{cases} \frac{4\alpha_i |x - c_{x_{ii}}| \sin \gamma_{ij}}{\rho_{ii}^2 \sqrt{\lambda_{ij}}} & \text{if } |x - c_{x_{ii}}| \leq \rho_{ii}/2 \text{ and } |\gamma_{ij}| \leq \pi/2, \\ -\frac{4\alpha_i |x - c_{x_{ii}}| \sin \gamma_{ij}}{\rho_{ii}^2 \sqrt{\lambda_{ij}}} & \text{if } |x - c_{x_{ii}}| \leq \rho_{ii}/2 \text{ and } |\gamma_{ij}| > \pi/2, \\ 0 & \text{if } |x - c_{x_{ii}}| > \rho_{ii}/2, \end{cases} \quad (22)$$

$$\frac{\partial V_i^k}{\partial \lambda_{ij}^k} = -\frac{\alpha_i |\cos \gamma_{ij}|}{2\sqrt{\lambda_{ij}^3}}. \quad (23)$$

4. Controller structure

This section presents a velocity controller for the lead car and a gap controller that keeps the follower cars at a constant distance from the leader.

4.1. Velocity controller

In a platoon each car tries to travel at the desired platoon velocity and maintain the correct spacing. The leader car chooses the desired platoon velocity. When the platoon travels at a constant velocity each car uses its own velocity controller to maintain the desired platoon velocity. These systems use the velocity and acceleration data that the car measures. The system output is the change in throttle angle.

The velocity controller for the *i*th car in the platoon has two inputs:

$$\Delta v_i(t) = v_{\text{platoon}} - v_i(t), \quad (24)$$

$$a_i(t) = a_i(t). \quad (25)$$

The output is the change in throttle angle $\partial_{\text{throttle}}$. So the fuzzy system defines the map $F: R^2 \rightarrow R$. One fuzzy rule is IF ($a_i(t)$ is zero (ZE)) AND ($\Delta v_i(t)$ is medium negative (MN)) THEN ($\partial_{\text{throttle}}$ is medium negative (MN)). The velocity difference and the acceleration each have 7 if-part fuzzy sets. The number of fuzzy rules for the velocity controller is $7 \times 7 = 49$. Figs. 6 and 7 show the fuzzy sets and rules for this controller [6].

4.2. Gap controller

Fig. 8 shows the block diagram of the fuzzy gap controller. It consists of a throttle controller and a brake controller. The gap controller maintains a constant distance between vehicles. The gap controller for platoon followers uses the differences in acceleration and velocity between cars and the distance error to achieve a constant gap. The distance error $\Delta d_i(k)$ is the difference between the desired gap between the cars and the actual gap. A range-finding system on each car in the platoon measured the distance between the cars. The inputs to the gap controller for the throttle in the *i*th car are

$$\Delta d_i(k) = d_{\text{desired}} - d_i(k), \quad (26)$$

$$\Delta v_i(k) = v_{i-1}(k) - v_i(k), \quad (27)$$

$$\Delta a_i(k) = a_{i-1}(k) - a_i(k). \quad (28)$$

So the fuzzy system defines the map $F: R^3 \rightarrow R$. One fuzzy rule is IF ($\Delta a_i(k)$ is ZE) AND ($\Delta v_i(k)$ is

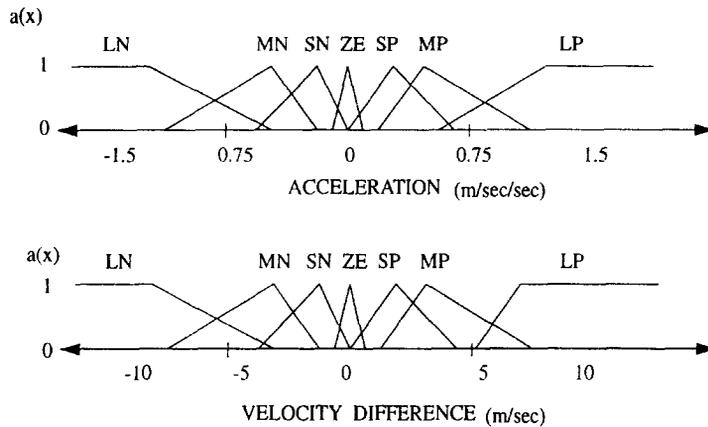


Fig. 6. Fuzzy sets for velocity controller.

| | | VELOCITY DIFFERENCE | | | | | | |
|--------------|----|---------------------|----|----|----|----|----|----|
| | | LP | MP | SP | ZE | SN | MN | LN |
| ACCELERATION | LN | EP | LP | LP | MP | MP | MP | ZE |
| | MN | LP | MP | MP | MP | ZE | ZE | MN |
| | SN | LP | MP | SP | SN | SN | SN | MN |
| | ZE | LP | MP | SP | ZE | SN | MN | LN |
| | SP | MP | SP | SP | SN | SN | MN | LN |
| | MP | MP | SP | SN | MN | MN | LN | EN |
| | LP | ZE | SN | MN | MN | LN | LN | EN |

Fig. 7. Fuzzy rules for velocity controller.

MN) AND ($\Delta d_i(k)$ is ZE) THEN ($\partial_{throttle}$ is MP). Figs. 9–11 show the fuzzy set values for the gap controller fuzzy variables Δd_i , Δv_i , and Δa_i .

The distance error and the velocity difference each have 7 if-part fuzzy sets. The number of rules for the throttle controller is $7 \times 7 \times 3 = 147$. These rules let a platoon maintain the desired gap.

The gap controller had only 3 fuzzy sets of acceleration difference as shown in Fig. 11. More acceleration sets would better predict the car response and give smoother control. But more sets result in a larger rulebase. We implemented the acceleration

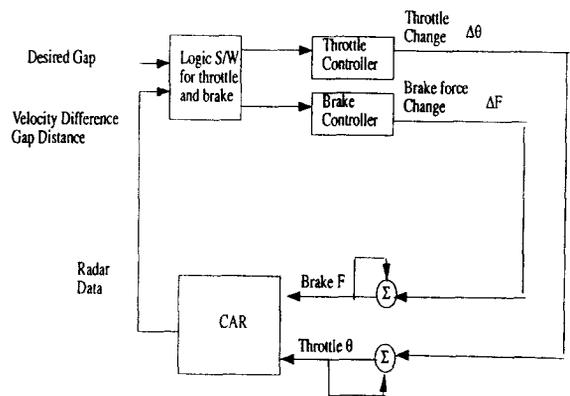


Fig. 8. Block diagram of the fuzzy gap controller that uses the brake and throttle.

input by using the estimated acceleration difference as described below. Figs. 12 and 13 show the control surfaces for different values of the distance error Δd_i .

The throttle actuator had a mechanical delay of 0.25 s. Closed-loop systems with time delays in the loops tend to have more stability problems than systems without delays [18]. Our controller used the acceleration data to predict the car’s motion to compensate for this delay. The vehicle sensors did not measure the acceleration difference directly. We can estimate the acceleration by differentiating the velocity. But this method is susceptible to noise since even small changes in velocity make the

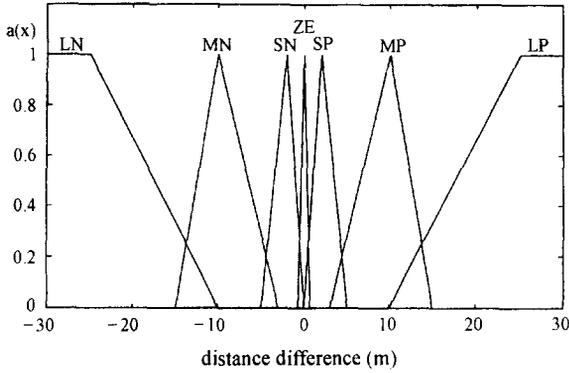


Fig. 9. If-part fuzzy set functions for distance difference input for the gap controller. The narrower sets near ZERO give finer control near the desired or equilibrium position.

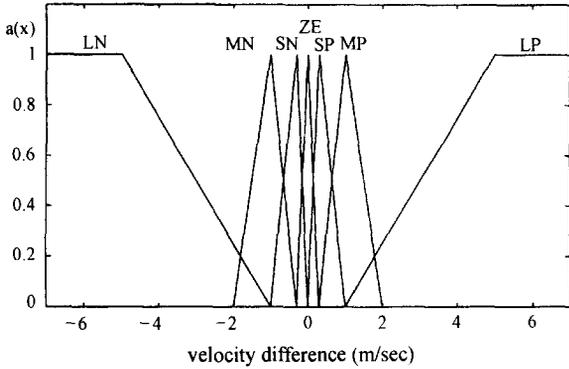


Fig. 10. If-part fuzzy set functions for velocity difference input for the gap controller.

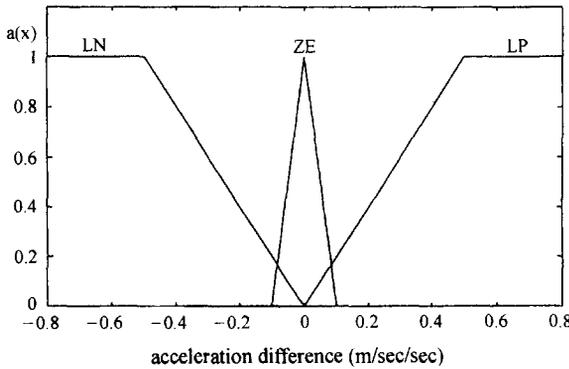


Fig. 11. If-part fuzzy set functions for acceleration difference input for the gap controller.

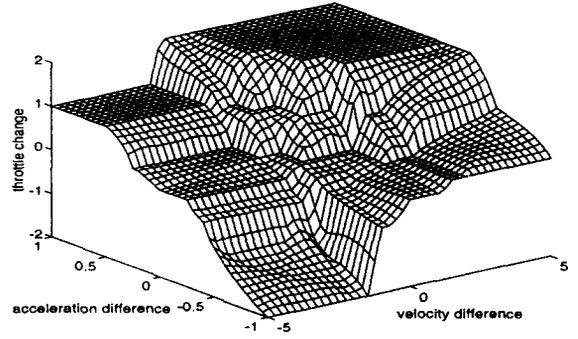


Fig. 12. Gap control surface when the distance difference is large negative (LN). Acceleration difference is m/s^2 , velocity difference is m/s , and throttle change unit is 0.02 per sampling time.

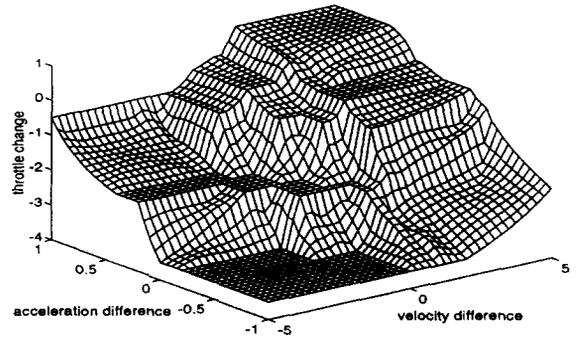


Fig. 13. Gap control surface when the distance difference is large positive (LP).

acceleration difference change greatly. It also creates high frequency terms of throttle change.

We instead approximate the acceleration input in (28) as the difference of the velocity measurements:

$$\Delta a_i(k) = \text{sign}(\Delta v_i(k) - \Delta v_i(k - 1))c, \tag{29}$$

where constant c depends on the sampling time. $\Delta a_i(k)$ can take only the 3 values $-c$, 0, or c . This approximation prevents the acceleration difference from drifting in the presence of noise. We used 0.05 s for the sampling time and $c = 20 \times 0.03048 = 0.6096$.

The output of the throttle controller in the i th car is the change in throttle angle $\Delta\theta_i(k)$. The input to the car is $\theta_i(k)$:

$$\theta_i(k) = \theta_i(k - 1) + \Delta\theta_i(k), \tag{30}$$

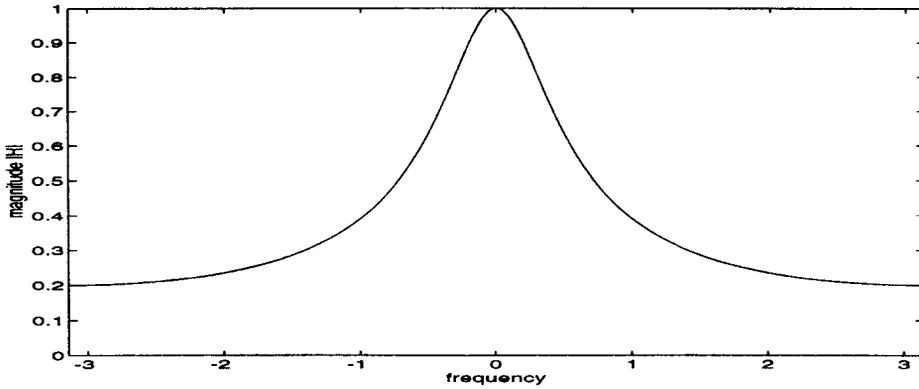


Fig. 14. Frequency response of the low-pass filter $H_{LP}(z)$.

where $\theta_i(k - 1)$ is the prior input to the car. A low-pass filter $H_{LP}(z)$,

$$H_{LP}(z) = \frac{\frac{1}{3}}{1 - \frac{2}{3}z^{-1}}, \tag{31}$$

smooths $\theta_i(k)$. So (31) gives

$$\theta_i^{LP}(k) = \frac{2}{3}\theta_i^{LP}(k - 1) + \frac{1}{3}\theta_i(k), \tag{32}$$

where $\theta_i^{LP}(k)$ is the filtered output. Fig. 14 shows the frequency response of our filter $H_{LP}(z)$. Putting (30) into (32) gives

$$\theta_i^{LP}(k) = \theta_i^{LP}(k - 1) + \frac{1}{3}\Delta\theta_i(k) \tag{33}$$

since $\theta_i(k - 1)$ in (30) becomes $\theta_i^{LP}(k - 1)$. We stored the fuzzy throttle controller as a look-up table based on the fuzzy sets and rules [8]. We scaled the output of the look-up table by $\frac{1}{3}$ to decrease round-off errors.

4.3. Brake controller

The gap controller can also use the brakes. The fuzzy brake controller outputs change in brake actuator level. It was 513 levels from 0 to 512. Then a brake model converts this level into a change in brake force for the simulation. Fig. 15 shows the block diagram of the brake controller. There are two inputs to the brake controller for the i th car:

$$\Delta d_i(k) = d_{\text{desired}} - d_i(k), \tag{34}$$

$$\Delta v_i(k) = v_{i-1}(k) - v_i(k). \tag{35}$$

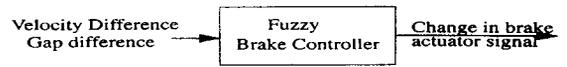


Fig. 15. Block diagram of brake controller.

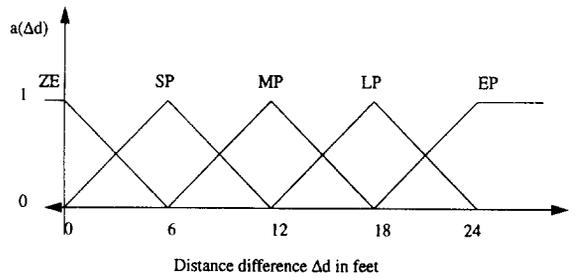


Fig. 16. If-part fuzzy set functions for distance difference input to the brake controller.

Figs. 16 and 17 show the fuzzy set values for the brake controller fuzzy variables Δd_i and Δv_i . One brake rule is IF Δd is Medium Positive (MP) and Δv is Medium Negative (MN) THEN the change in brake actuator is Medium Small (MS). Fig. 18 shows the 6 then-part sets for the output fuzzy variable Δb_i . The sets do not have the same area and thus do not have the same V_j terms in the SAM equation (2) for $F(x)$.

The brake controller has $5 \times 5 = 25$ fuzzy rules. Fig. 19 shows the fuzzy rules for the brake controller. Nine fuzzy sets quantize the fuzzy variables Δd_i and Δv_i .

- EP: Extreme Positive
- LP: Large Positive

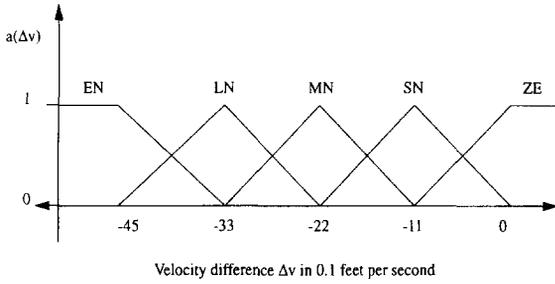


Fig. 17. If-part fuzzy set functions for velocity difference input to the brake controller.

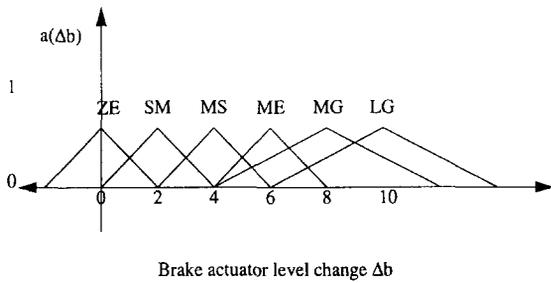


Fig. 18. Then-part fuzzy set functions for the brake actuator change. The fuzzy system F uses just the centroids c_j and the areas or volumes V_j of the 6 then-part sets to compute the output $F(x)$. Appendix A gives the details of this computation.

| | | VELOCITY DIFFERENCE | | | | |
|------------------------------------------------|----|---------------------|----|----|----|----|
| | | ZE | SN | MN | LN | EN |
| D I S T A N C E | ZE | ZE | ZE | ZE | SM | MS |
| | SP | ZE | ZE | SM | MS | ME |
| | MP | ZE | SM | MS | ME | ML |
| | LP | SM | MS | ME | ML | LG |
| | EP | MS | ME | ML | LG | LG |
| D I F F E R E N C E | | | | | | |

Fig. 19. Fuzzy rules for the brake controller. The if-part fuzzy sets are Δv and Δd . The then-part sets give the change Δb in the brake actuator signal.

- MP: Medium Positive
- SP: Small Positive
- ZE: Zero
- SN: Small Negative

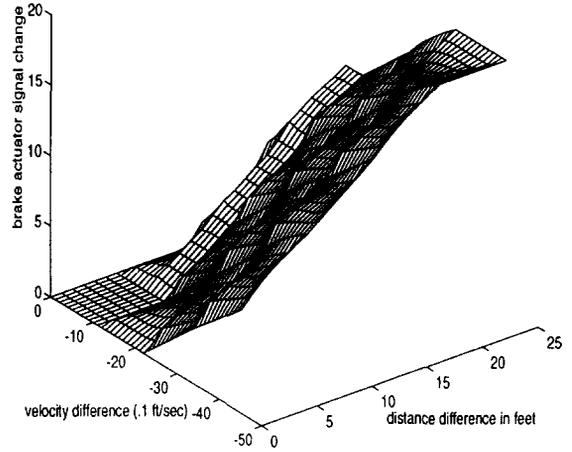


Fig. 20. Control surface for fuzzy brake controller. The output unit is 0.15° change for the brake pedal.

- MN: Medium Negative
- LN: Large Negative
- EN: Extreme Negative

Fig. 20 shows the control surface of the brake controller. The brakes are on only when the distance difference is positive (when the follower car is too close) and the relative velocity is negative (when the follower car goes faster than the lead car).

We combined the throttle and brake outputs using a logic switch that transitions between the brake and the throttle. Fig. 21 shows how the system used the brake and the throttle for different distance errors and velocity differences. The brake region shows when the brake is on and the throttle is off. The brake comes on only when the car is closer than the desired distance and the follower car goes faster than the car ahead.

A “neutral region” [12] can help avoid frequent transitions between the throttle and brake fuzzy systems. The brake control signal does not change and the throttle is off when the inputs are in the neutral region. Fig. 21 shows the neutral region for the fuzzy brake-controller. The neutral region covers small values of the inputs Δv and Δd . The neutral region is that part of the control surface in Fig. 20 that equals zero.

The hardware on the test car had limited memory and used only integer operations. It stored the fuzzy gap controller as a look-up table based on the

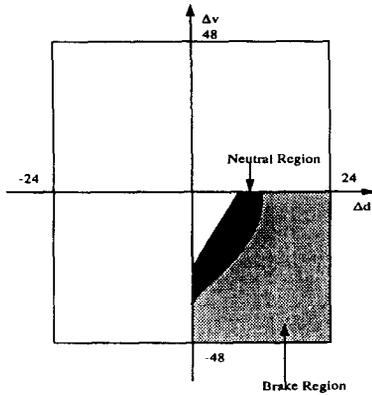


Fig. 21. Domain of the look-up table for the fuzzy gap controller.

fuzzy sets and rules for throttle and brake control. Fig. 21 shows the domain for the look-up table. The rectangle shows the control region for the gap controller look-up table. The white region shows the throttle system. The shaded region shows the brake system. And the black region shows the neutral region. The horizontal axis gives the distance difference in feet (ft). The vertical axis gives the velocity difference in $\frac{1}{10}$ ft/s. We thresholded all inputs outside these limits to the maximum or minimum values of the domain. The fourth quadrant shows the brake look-up table. One fuzzy rule for this case has the form IF ($\Delta v_i(k)$ is MN) AND ($\Delta d_i(k)$ is MP) THEN (\hat{v}_{brake} is MS).

The look-up table for the throttle system had $3 \times 49 \times 97 = 14\,259$ entries. The distance difference varied from -24 ft to 24 ft in 1 ft units and had 49 entries. The velocity difference had 97 entries as it varied from -4.8 ft/s to 4.8 ft/s in $1/10$ ft/s units. The look-up table for the brake system had $25 \times 49 = 1225$ entries since it applied only in the fourth quadrant.

5. Car models and sensor system

We used two car models to test our fuzzy velocity and gap controllers. The first model was a second-order car model that we used to check our controller design and to test the fuzzy rules. The second model was a nonlinear “validated” longitu-

dinal car model. The Ford Motor Company designed this model and we used it to test our controllers.

5.1. Car models

The second-order car model [19] had two equations of motion:

$$m_i a_i = m_i \xi_i - K_d v_i^2 - d_m - m_i g \sin \beta_i, \tag{36}$$

$$\dot{\xi}_i = -\frac{\xi_i}{\tau_i(v_i)} + \frac{u_i}{m_i \tau_i(v_i)}. \tag{37}$$

The force law (36) comes from Newton’s second law of motion $F = ma$. The term $m_i \xi_i$ is the tractive engine force that the wheels apply to the road. The variables a_i and v_i stand for acceleration and velocity. $K_d v_i^2$ and d_m give the aerodynamic and mechanical drag forces. m_i is the total mass of the car and cargo and τ_i is the engine time lag. The term $m_i g \sin \beta_i$ gives the acceleration due to gravity. β_i is the inclination of the hill from the horizontal and g is the acceleration due to gravity of 9.8 m/s^2 . The throttle angle u_i is the input to the system and changes the “jerk” or rate of acceleration. We used the coefficients [6] $\tau_i = 0.2$ seconds, $K_d = 0.44 \text{ kg/m}$, and $d_m = 352 \text{ kgm/s}^2$.

Fig. 22 shows the basic subsystems of the validated longitudinal car model that we used to design and test our controllers. Each block of the

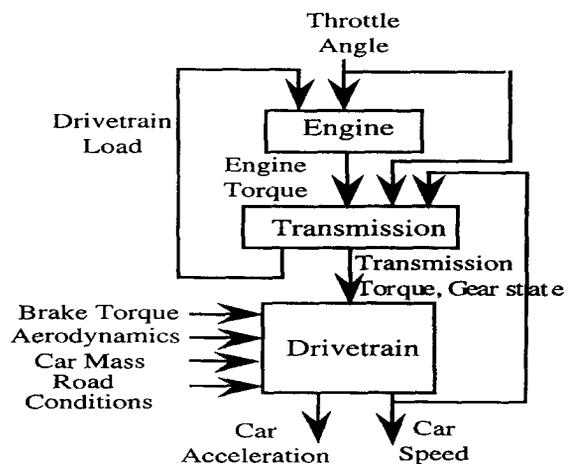


Fig. 22. Validated car model from the Ford Motor Company.

model is a car subsystem. The engine torque is the output of the engine subsystem. The engine torque is a nonlinear function of the air/fuel ratio, the exhaust gas recirculation, the cylinder total mass charge, the spark advance, the engine speed, the drivetrain load, and the throttle angle [12]. The output of the transmission subsystem is the transmission torque and gear state. The drivetrain system computes the car's velocity and acceleration based on the road conditions and the car loading. Ford Motor Company [12] "validated" or tested and tuned this proprietary math model. The inputs to this model are the throttle angle and the brake force. The outputs are the car's speed and acceleration.

Each car has a radar system above its front bumper. The radar measures the distance and the velocity difference between the computer-controlled car and the car ahead.

5.2. Brake models

We modeled the brake to test the fuzzy brake controller. The Ford-tuned car model in Fig. 22 needs an input brake torque. The brake controller gives as output a bit value to the brake actuator.

We modeled the brake using brake data from VORAD Incorporated. The test car was a Lincoln Town Car. The data gave the deceleration based on the brake actuator bit input value and the car velocity. Newton's second law $F = ma$ gave the total force that equals the input bit value for the mass m of 4031 lb.

The nonlinear validated longitudinal car model gave a total drag force Y in lb:

$$Y = 50 + 0.819 \times V + 0.0192 \times V^2. \quad (38)$$

V is the car velocity in miles per hour (mph). Subtracting the drag force from the total force gives the brake force that corresponds to the brake actuator input bit value:

$$F_b = F - Y - D. \quad (39)$$

F_b is the brake force and D is the force from an external disturbance such as a hill. We interpolated between data points to get the curve in Fig. 23. It shows the brake force with respect to the actuator

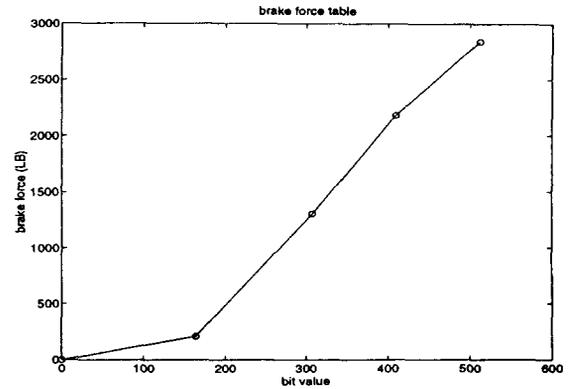


Fig. 23. Brake force as a function of the actuator bit value.

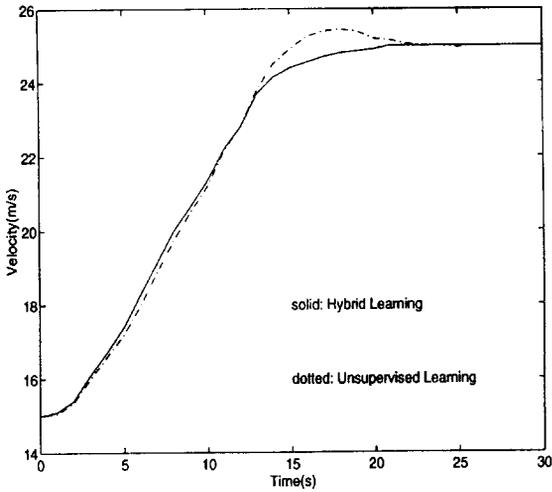
bit value using $F = ma$ and (38) with brake data from the Lincoln Town Car.

6. Simulation and test results

6.1. Learning fuzzy rules

We used the hybrid learning system described in Section 3 to learn the fuzzy sets and rules for the velocity controller. Unsupervised learning gave the first set of rules. Then we tuned these rules to improve the controller's response. The training data came from the car model in (36) and (37) [19]. The leader velocity controller in [6] gave 7500 training samples in 200 trajectories for a sports utility car. The training vectors $(a, \Delta v, \partial_{\text{throttle}})$ defined points in the 3-D input-output space. Unsupervised ellipsoid covariance learning clustered the data and computed its local statistics. The adaptive vector quantization system had 450 synaptic vectors or local pattern classes as discussed in Appendix B. The sum of the ellipsoid projections onto each axis of the state space gave a histogram of the density of the pattern classes. We chose 7 if-part subsets of each of the input axes. The center of each fuzzy set matched a peak in the histogram.

We partitioned the state space into a grid of rule patches. To find the rules we counted the number of synaptic vectors in each cell. Clusters of synaptic vectors in fuzzy rule cells defined the rules [14].



Appendix B gives the details of this unsupervised *product space* clustering.

Then supervised learning tuned the rules to minimize the mean-squared error for the training data. There were 49 rules. The supervised system used 30 000 cycles to tune the 49 rules. Fig. 24 compares the platoon velocity for the lone unsupervised and hybrid controllers. The desired velocity was 25 m/s. The hybrid controller accelerated faster than did



Fig. 24. Comparison of the velocity controller performance after lone unsupervised learning and after hybrid learning with unsupervised clustering and supervised gradient descent. The hybrid controller gave a faster response and had no overshoot.

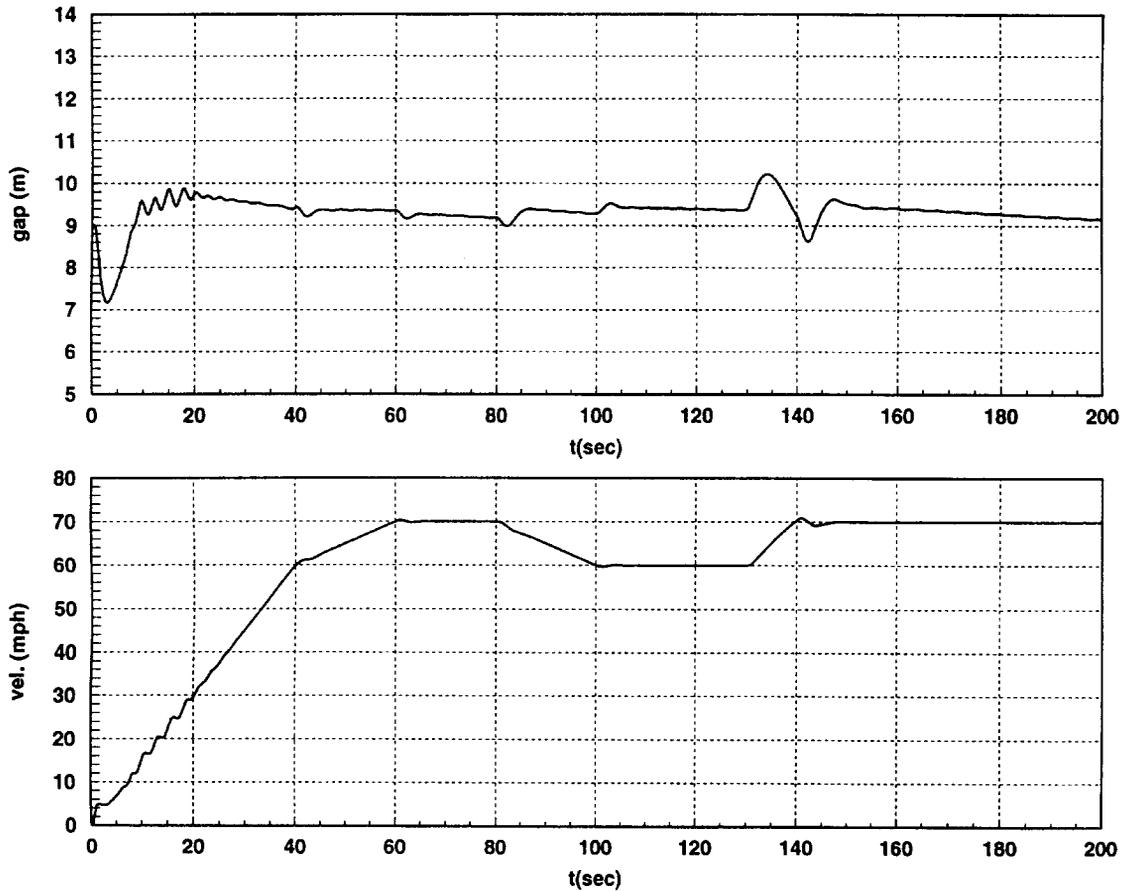


Fig. 25. Simulation results for the second follower car in a three-car platoon. The car changes speed in the presence of hills. It tends to maintain the desired gap of 9 m between it and the platoon leader. The throttle actuator had a mechanical delay of 0.25 s.

the unsupervised learning system. The hybrid controller had no overshoot at the desired speed.

6.2. Gap controller with throttle only

The hardware on the test car had limited memory and used only integer operations. We stored the fuzzy controller as a look-up table based on the fuzzy sets and rules in Section 4.2. We simulated small platoons over a range of velocity changes. The throttle actuator had a mechanical delay of 0.25 s. We used the validated car model Fig. 22 for these simulations. Figs. 25 and 26 show the results for a three-car platoon that changes velocity due to terrain changes such as hills. The desired gap dis-

tance was 9 m. The follower cars maintained the desired gap distance except for the transients at the start of the simulation.

We also simulated cases where additive measurement noise corrupted the sensor input $\Delta v_i(k)$. The single-to-noise ratio (SNR)

$$SNR = 10 \log_{10} \frac{\sigma_x^2}{\sigma_n^2} \text{ db} \tag{40}$$

measured the uncertainty in the sensor data. Here σ_x^2 and σ_n^2 stand for the variance of the velocity difference signal Δv_i and the variance of the measurement noise. Figs. 27 and 28 show that the fuzzy throttle system gave robust control in the

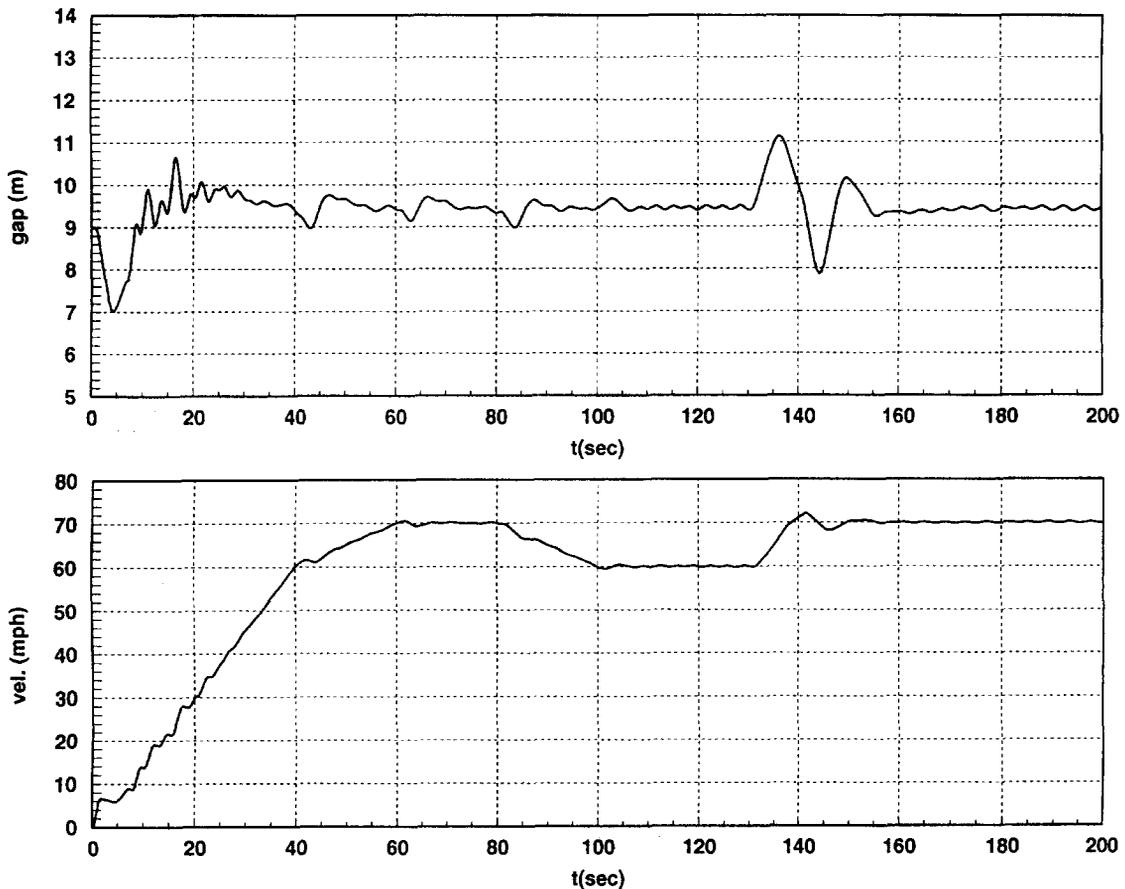


Fig. 26. Simulation results for the third follower car in a three-car platoon. The third car tends to maintain the desired gap of 9 m between it and the second car but it does so with more variability.

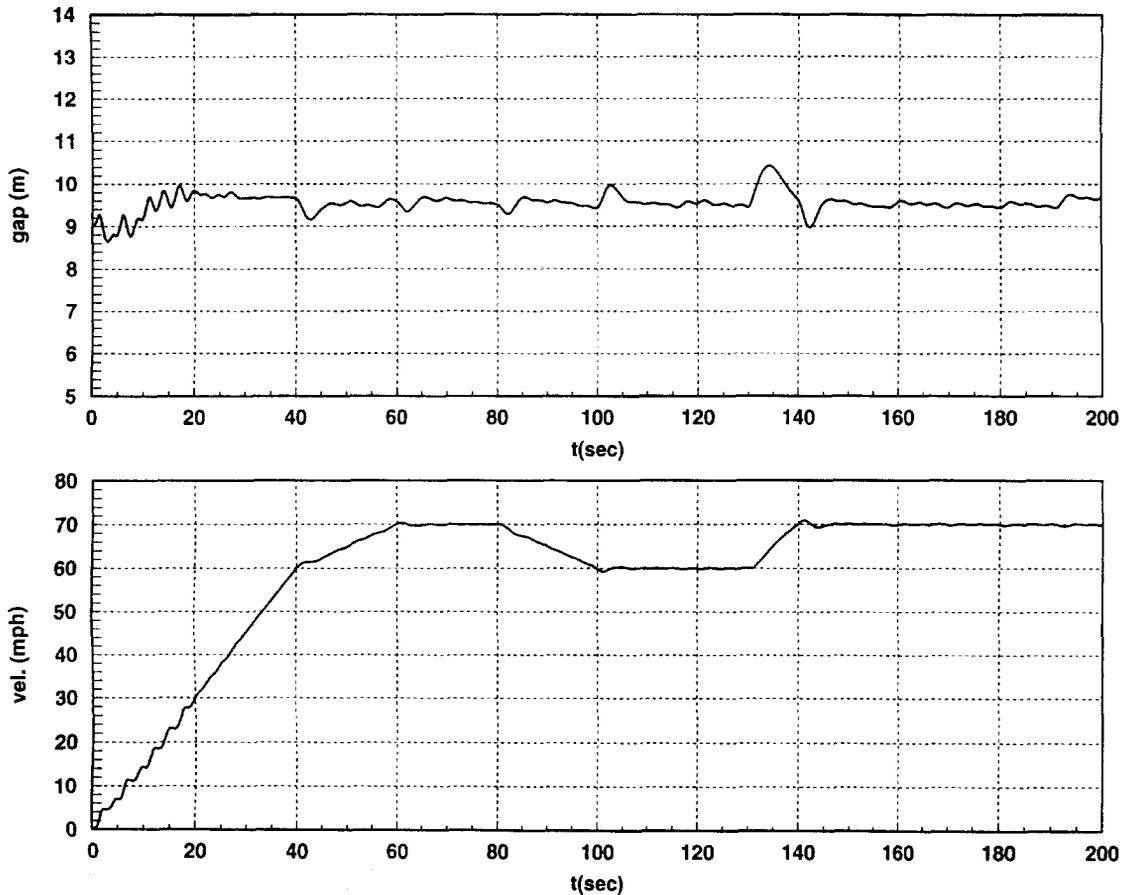


Fig. 27. Simulation results for the follower car in a two-car platoon when the signal-to-noise ratio is 27 db. The gap error due to noise never exceeds 1 m.

presence of noise. The noise had little effect on the velocity of the follower car in the two-car platoon. The noise did make the gap jiggle but never by more than 1 m.

6.3. Roadway tests for throttle controller

We tested the gap controller on highway I-15 in Escondido, California. We put our controller in a Lincoln Town Car from VORAD Incorporated. The follower car got data from the radar system on the front of the car.

The radar measured the distance and the velocity difference between the computer-controlled car and the car ahead. The radar tracked the car ahead and

had a measurement delay of 0.05 s. Fig. 29(a) shows the follower car gap as the platoon accelerated. Fig. 29(b) shows the closing rate between the cars. Fig. 29(c) shows the throttle value as the car accelerated. The desired gap was 125 ft. The follower car dropped back because the initial gap was too short.

The platoon went up and down hills in the second test. The desired gap was again 125 ft. The follower car dropped back as the platoon started up the hill. Fig. 30(a) shows the gap distance as the platoon went up a hill. The follower dropped back and then moved to the right gap. Fig. 30(b) shows the closing rate between the cars. The spike at 15 s occurred when the radar sensor briefly lost the lead

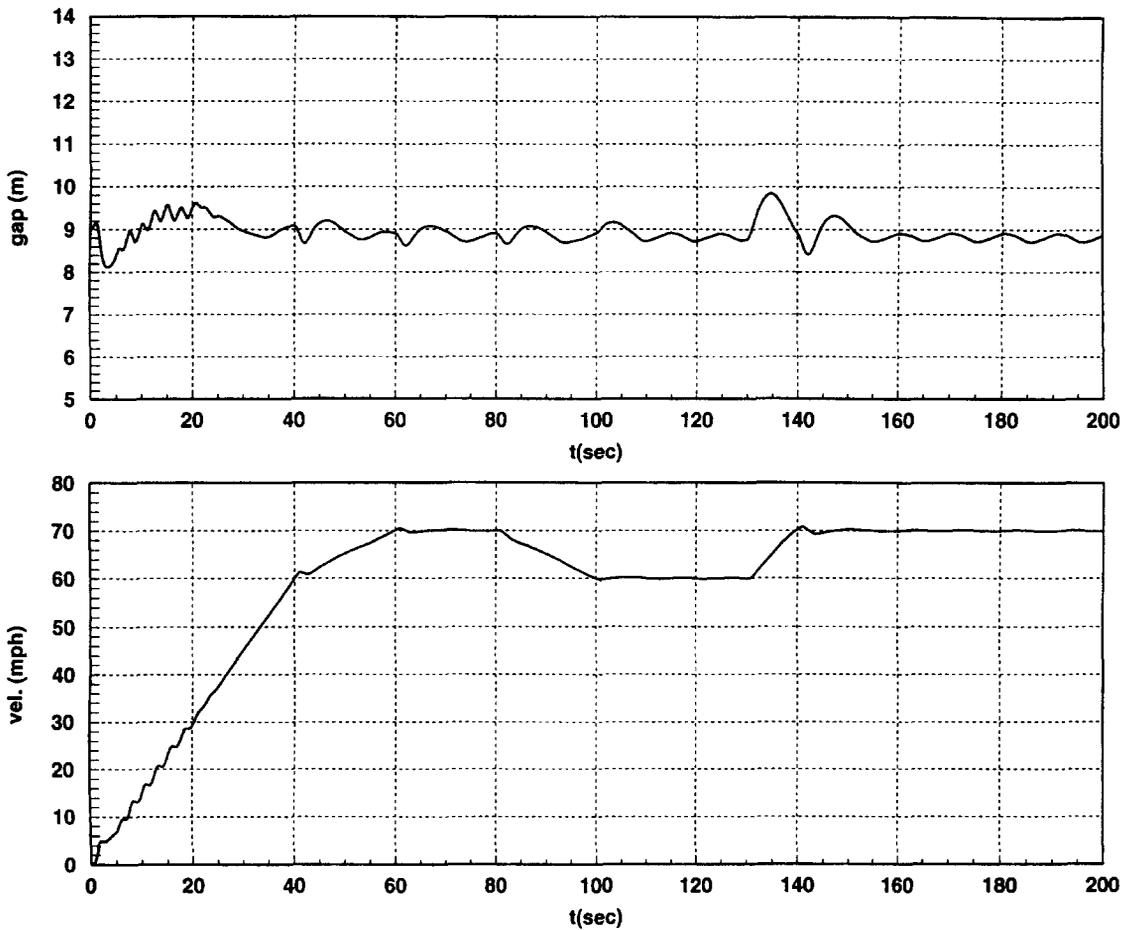


Fig. 28. Simulation results for the follower car in a two-car platoon when the signal-to-noise ratio is 11 db. The additive noise has less effect on the car's velocity than on the gap it tries to maintain with the lead car.

car in the platoon. The follower car maintained a constant throttle until the sensor detected a new target. Fig. 30(c) shows the throttle value as the car went up the hill.

6.4. Gap controller with throttle and brake

The gap controller must use the brakes to slow the car if the engine torque is not sufficient. We simulated cases where braking can avoid a collision as when the platoon moves up and down hills or when the platoon slows down.

We simulated hills as external disturbances. Fig. 31 shows the leader car's velocity profile and

the external disturbances. These disturbances correspond to a 5% grade both uphill and downhill. Fig. 32 shows the simulation results with the throttle-only gap controller. The car cannot avoid a collision without braking due to the steep downhill grades.

Fig. 33 shows the simulation results for a gap controller that uses both the brake and throttle. The follower car applies the brakes so that it will not hit the leader. The gap decreases to 5 m before the car slows to the desired speed of 60 mph.

The follower cars also need brake control when the platoon slows down. Fig. 34 shows the simulation results without using a brake when the platoon

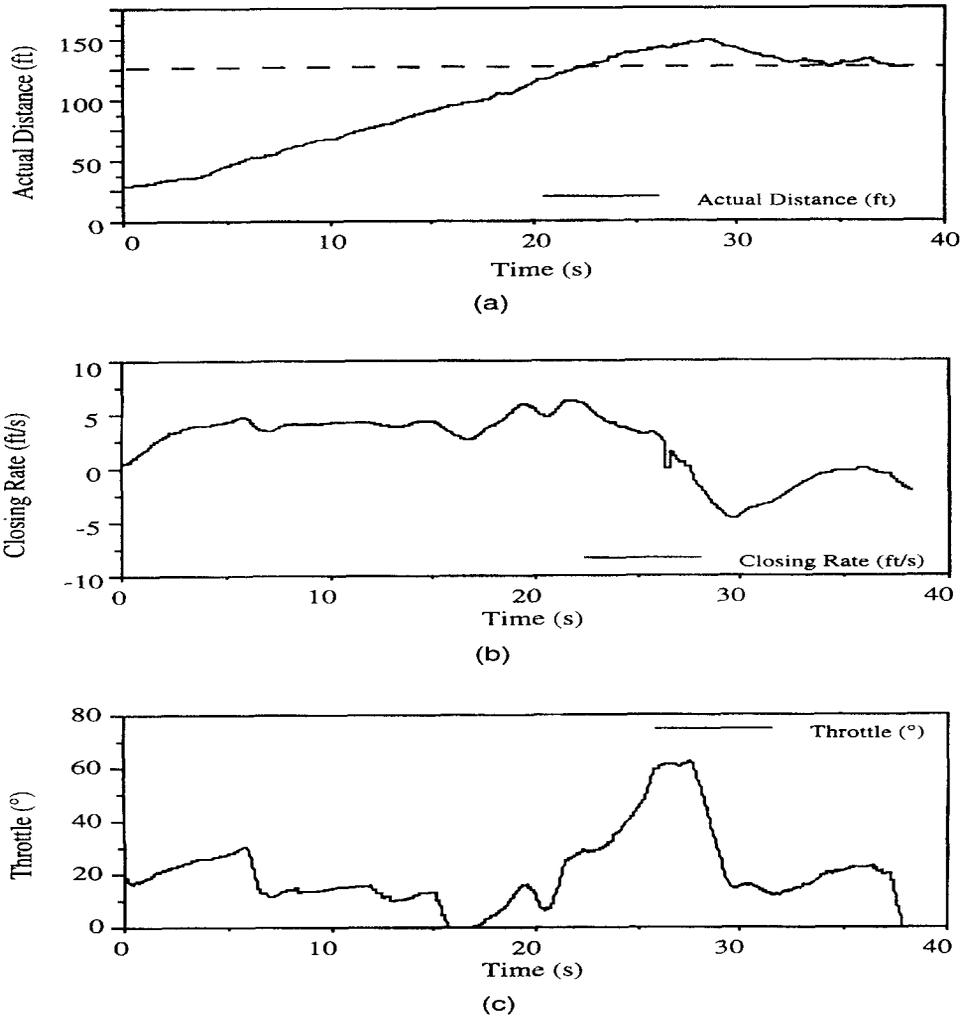


Fig. 29. Follower car data as a two-car platoon accelerates on the highway. (a) shows the gap. (b) shows the closing rate between the cars. (c) shows the throttle values for the follower cars. The follower car drops back because the initial gap was less than the desired gap of 125 ft. The radar had a measurement delay of 0.05 s.

decelerates. The leader car decelerates from 70 to 50 mph starting at $t = 160$ s. The leader car maintains 50 mph at $t = 170$ s. The deceleration at $t = 160$ s forced the follower car to brake to avoid a collision. Engine torque cannot slow the car down enough. Fig. 35 shows the simulation results of the combined throttle and brake system. The combined system avoids the collision and does not oscillate between the throttle and brake.

7. Conclusion

Additive fuzzy systems can control the velocity and the gap of cars in single-lane platoons. We used this controller to drive the smart car on the highway in a two-car platoon. We first designed and tested the controller using throttle only with a validated car model and then with a real car on highway I-15 in California. Then we added the

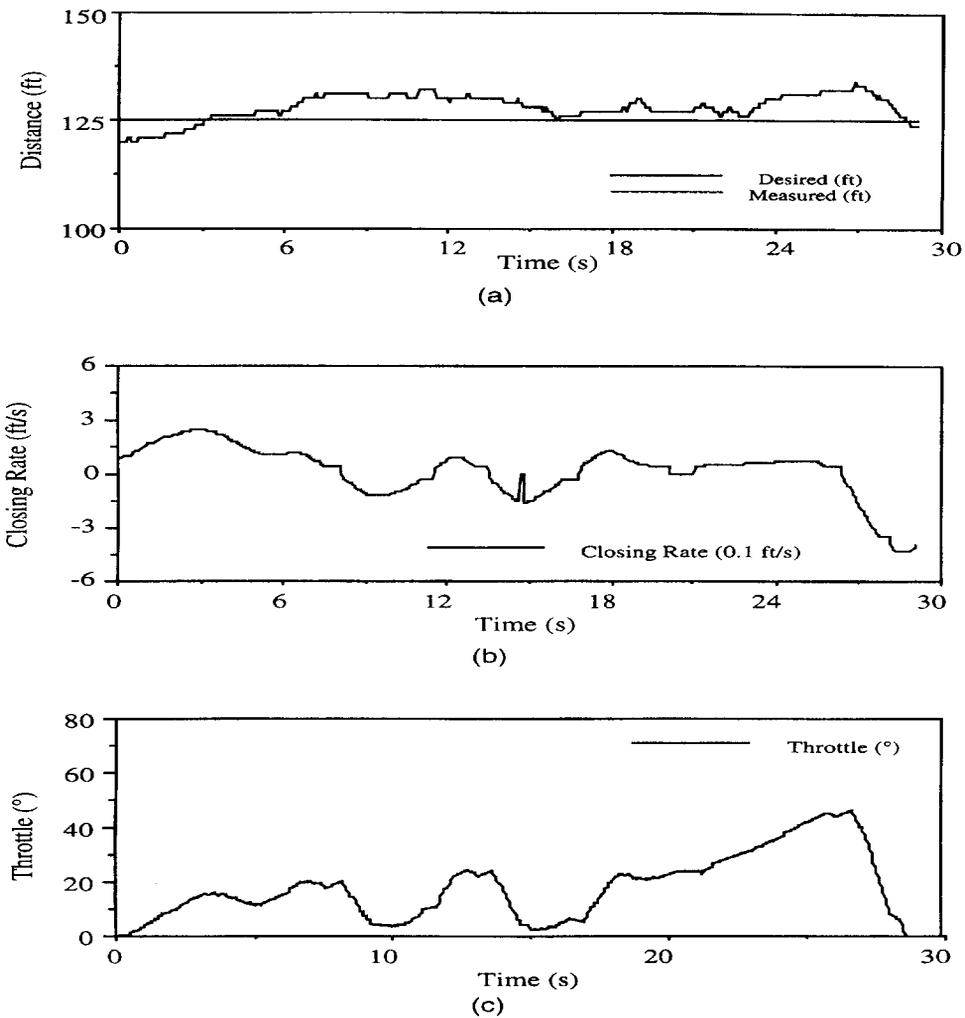


Fig. 30. Follower car data as a two-car platoon climbs a hill. (a) shows the gap. (b) shows the closing rate between the cars. The spike at 15 s shows where the radar sensor briefly lost the lead car in the platoon. (c) shows the throttle values for the follower cars as the platoon went uphill.

brake controller and simulated it. The next phase of the fuzzy platoon controller will add the steering controllers so the platoon can maneuver on the highway.

The controller worked well for coupled systems where a series of objects must track and predict the object in front of it. Networks of these controllers could control the rate of message or car traffic flow through electronic and physical intersections. The coupled system can differ. The distributed structure

of the fuzzy controller could apply to factory assembly lines or to robotic limb control.

Unsupervised ellipsoidal learning tuned the fuzzy rules and sets for cars of different sizes and engine types. This gives a new way to find a fuzzy system using only data from a human driver or other controller. Supervised learning further tuned the rules. Ellipsoidal learning can tune any control system if it has access to input–output data as in the control of many biological or economic processes.

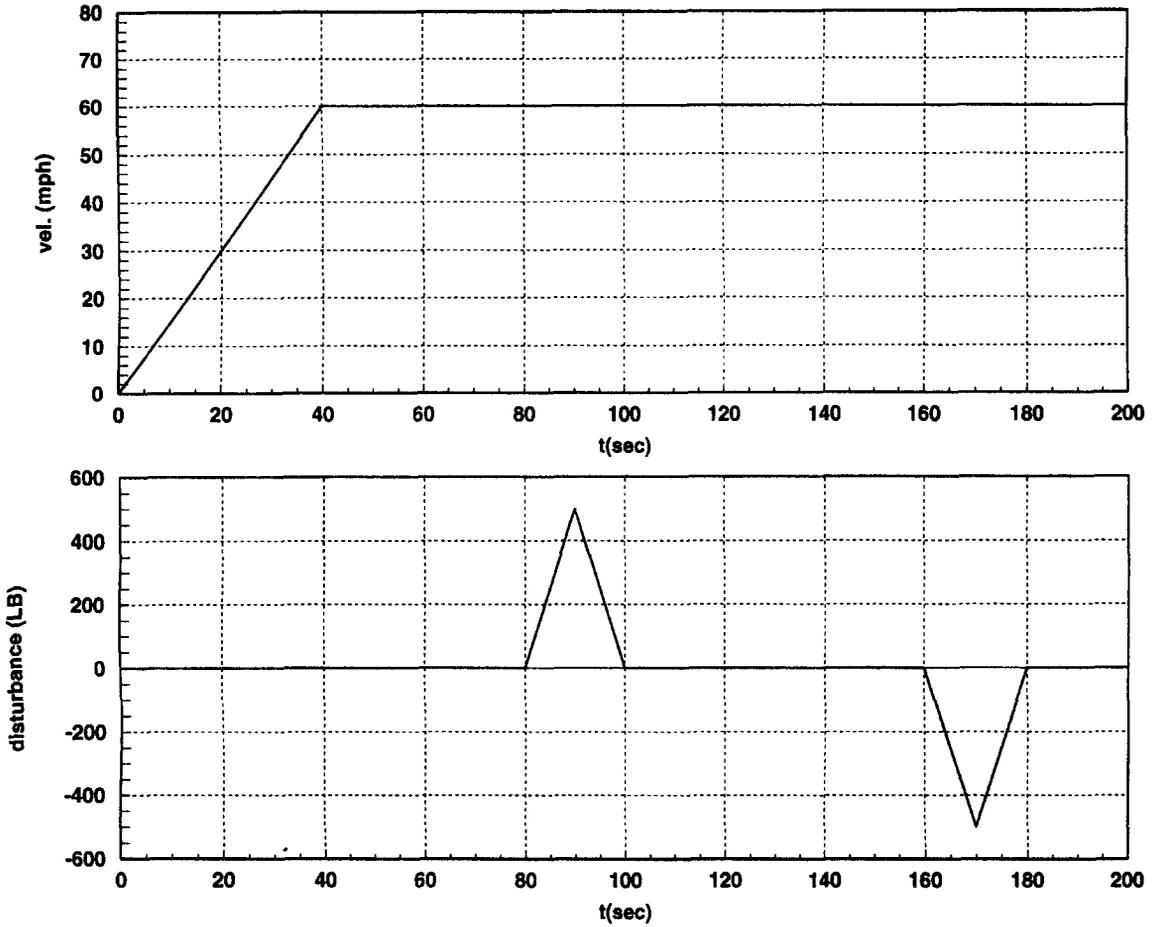


Fig. 31. Profile of the leader car in a two-car platoon. External forces model hills. The triangles model both a 5% grade uphill and downhill.

On-line adaptive fuzzy control systems with ellipsoidal learning can adapt the system over time as engine parameters and road conditions change. Future learning schemes may tune the ellipsoidal rules with techniques other than gradient descent or may use rules of other shapes that give a better function approximation.

Acknowledgements

The Caltrans PATH Program supported this research (Agreement #20695 MB). VORAD Incorporated provided the Lincoln Town Car and radar sensor for the gap controller test.

Appendix A: SAM theorem

Theorem. Suppose the fuzzy system $F: R^n \rightarrow R^p$ is a standard additive model (SAM): $F(x) = \text{Centroid}(B) = \text{Centroid}(\sum_{j=1}^m a_j(x)B_j)$. Then $F(x)$ is a convex sum of the m then-part set centroids:

$$F(x) = \frac{\sum_{j=1}^m a_j(x)V_j c_j}{\sum_{j=1}^m a_j(x)V_j} = \sum_{j=1}^m p_j(x)c_j. \tag{A.1}$$

The convex coefficients or discrete probability weights $p_1(x), \dots, p_m(x)$ depend on the input x through

$$p_j(x) = \frac{a_j(x)V_j}{\sum_{k=1}^m a_k(x)V_k}. \tag{A.2}$$

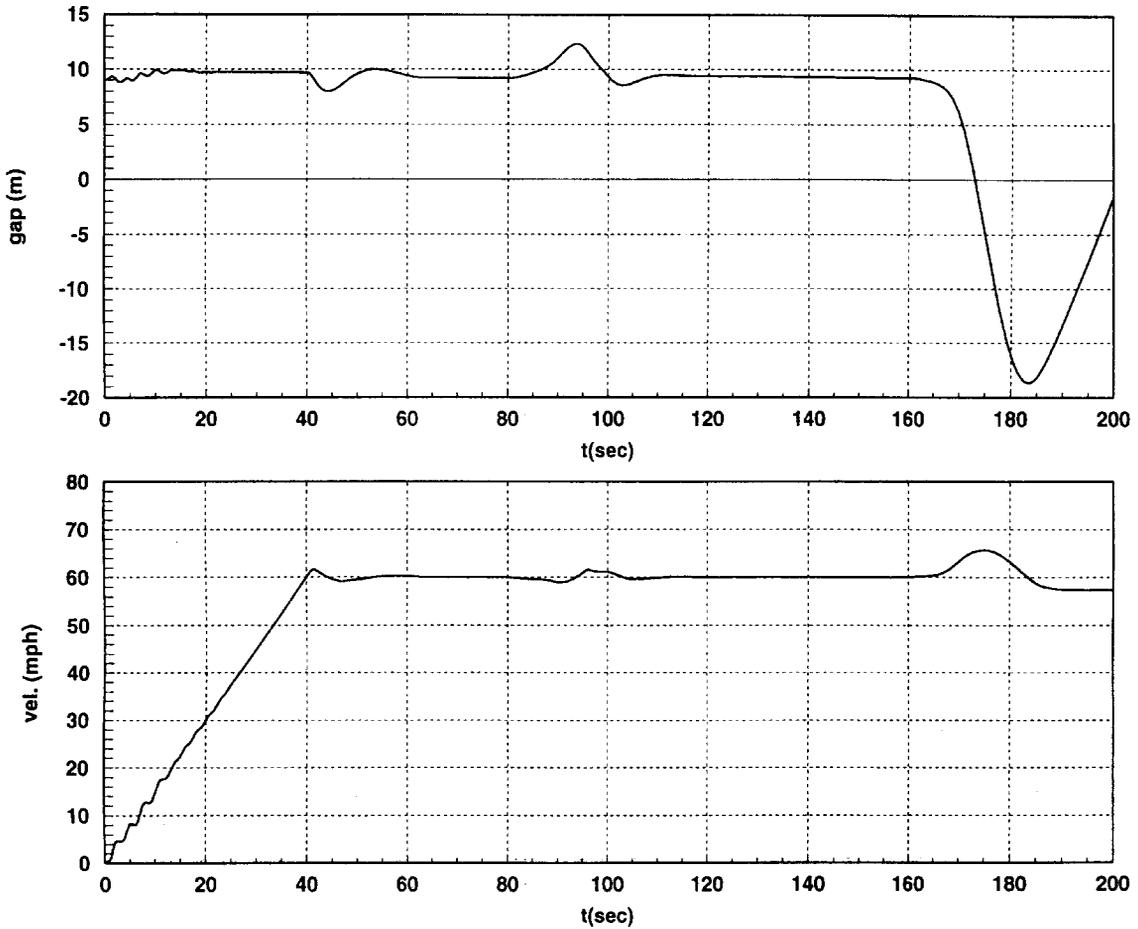


Fig. 32. Simulation results for the follower car using the throttle only for going uphill and downhill. The follower car cannot slow down enough to avoid hitting the lead car on the downhill grade ($t \approx 170$ s).

V_j is the finite positive volume (or area if $p = 1$) and c_j is the centroid of the then-part set B_j :

$$V_j = \int_{R^p} b_j(y_1, \dots, y_p) dy_1 \dots dy_p > 0, \tag{A.3}$$

$$c_j = \frac{\int_{R^p} y b_j(y_1, \dots, y_p) dy_1 \dots dy_p}{\int_{R^p} b_j(y_1, \dots, y_p) dy_1 \dots dy_p}. \tag{A.4}$$

Proof. There is no loss of generality to prove the theorem for the scalar-output case $p = 1$ when $F: R^n \rightarrow R^p$. This simplifies the notation. We need but replace the scalar integrals over R with the p -multiple or volume integrals over R^p in the proof to prove the general case. The scalar case $p = 1$

gives (A.3) and (A.4) as

$$V_j = \int_{-x}^x b_j(y) dy, \tag{A.5}$$

$$c_j = \frac{\int_{-x}^x y b_j(y) dy}{\int_{-x}^x b_j(y) dy}. \tag{A.6}$$

Then the theorem follows by expanding the centroid of B and invoking the SAM assumption $F(x) = \text{Centroid}(B) = \text{Centroid}(\sum_{j=1}^m a_j(x)B_j)$ to rearrange terms:

$$F(x) = \text{Centroid}(B) \tag{A.7}$$

$$= \frac{\int_{-x}^x y b(y) dy}{\int_{-x}^x b(y) dy} \tag{A.8}$$

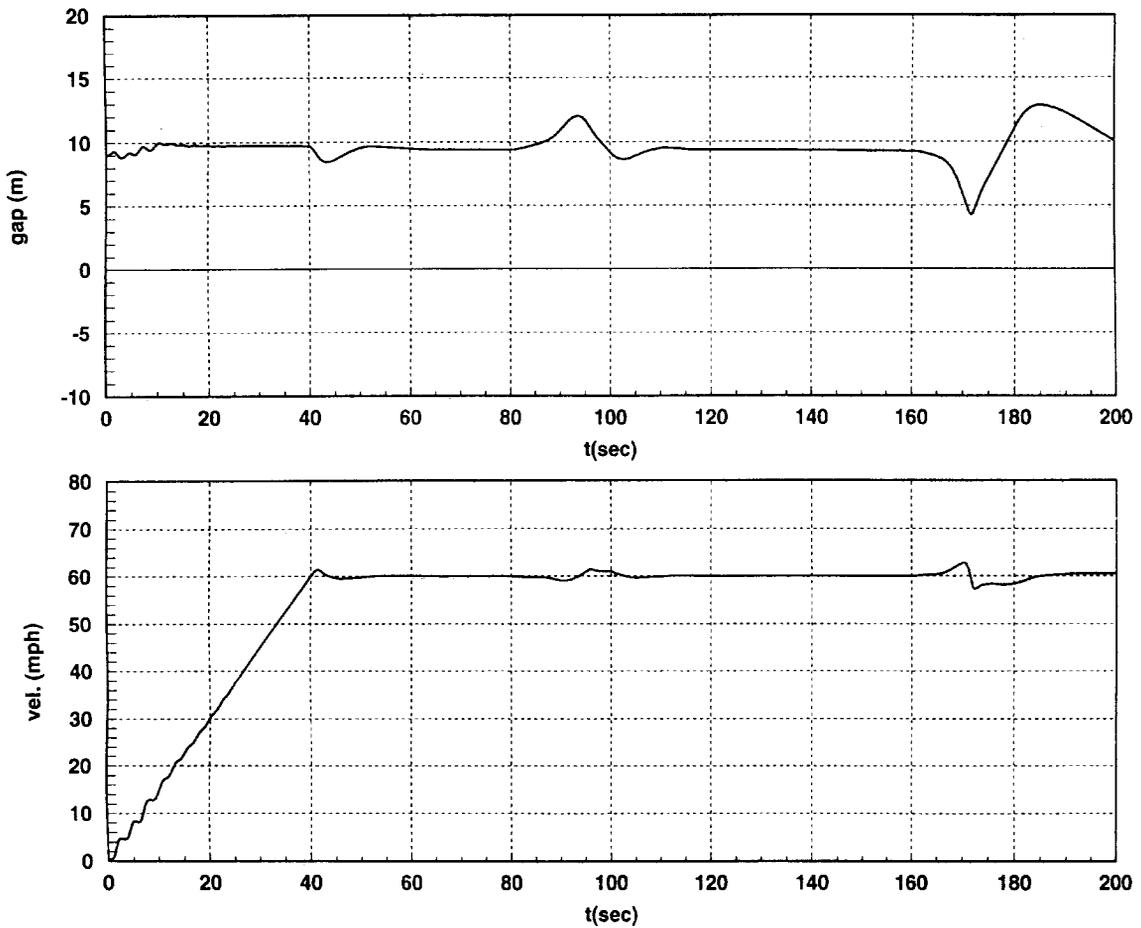


Fig. 33. Simulation results for the follower car with a gap controller that combines throttle and brakes. The cars follow the velocity and terrain profile in Fig. 24. The follower car approaches the leader but does not collide with it.

$$= \frac{\int_{-\infty}^{\infty} y \sum_{j=1}^m b'_j(y) dy}{\int_{-\infty}^{\infty} \sum_{j=1}^m b'_j(y) dy} \quad (\text{A.9})$$

$$= \frac{\int_{-\infty}^{\infty} y \sum_{j=1}^m a_j(x) b_j(y) dy}{\int_{-\infty}^{\infty} \sum_{j=1}^m a_j(x) b_j(y) dy} \quad (\text{A.10})$$

$$= \frac{\sum_{j=1}^m a_j(x) \int_{-\infty}^{\infty} y b_j(y) dy}{\sum_{j=1}^m a_j(x) \int_{-\infty}^{\infty} b_j(y) dy} \quad (\text{A.11})$$

$$= \frac{\sum_{j=1}^m a_j(x) V_j (\int_{-\infty}^{\infty} y b_j(y) dy) / V_j}{\sum_{j=1}^m a_j(x) V_j} \quad (\text{A.12})$$

$$= \frac{\sum_{j=1}^m a_j(x) V_j c_j}{\sum_{j=1}^m a_j(x) V_j} \quad \square \quad (\text{A.13})$$

We can further weight each rule with a scalar weight w_j to give

$$F(x) = \text{Centroid} \left(\sum_{j=1}^m w_j a_j(x) B_j \right) \quad (\text{A.14})$$

$$= \frac{\sum_{j=1}^m w_j a_j(x) V_j c_j}{\sum_{j=1}^m w_j a_j(x) V_j} \quad (\text{A.15})$$

The weight w_j has the same form as the area or volume weight V_j . We use $w_1 = \dots = w_m > 0$ in the paper. So the weighted sum (A.15) reduces to (A.1). The centroidal structure (A.7) and (A.8) of the fuzzy system shows that *all* centroidal fuzzy systems

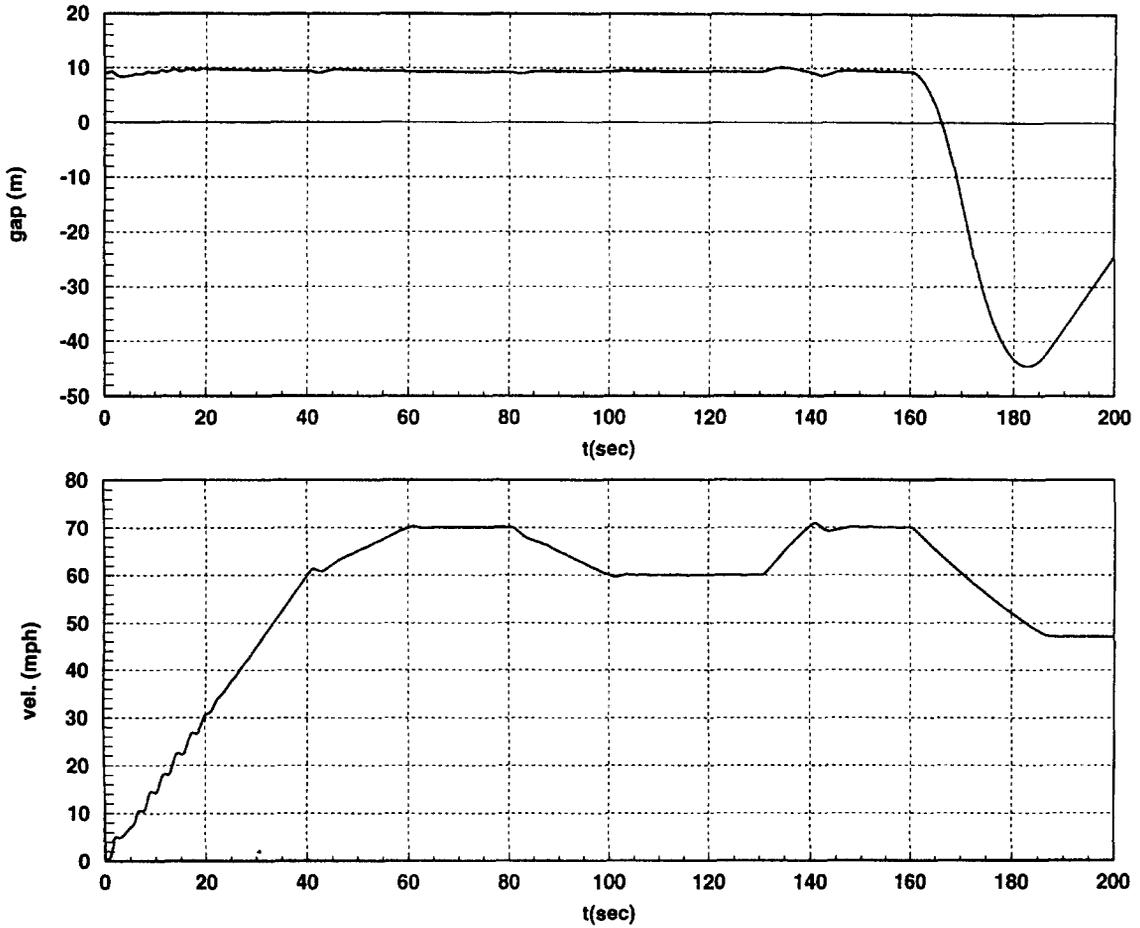


Fig. 34. Simulation results using throttle only for a lead car deceleration of $0.09g$. The follower car collides with the leader at $t \approx 165$ s.

compute a conditional expectation:

$$F(x) = \frac{\int_{-\infty}^{\infty} yb(x, y) dy}{\int_{-\infty}^{\infty} b(x, y) dy} \tag{A.16}$$

$$= \int_{-\infty}^{\infty} yp(y|x) dy \tag{A.17}$$

$$= E[Y | X = x] \tag{A.18}$$

since

$$p(y|x) = \frac{b(x, y)}{\int_{-\infty}^{\infty} b(x, y) dy} \tag{A.19}$$

defines a conditional probability density even though $b(x, y) > 1$ may hold for the integrable function $b \geq 0$.

The SAM ratio (A.1) reduces to Sugeno’s [21] “center of gravity” model

$$F(x) = \frac{\sum_{j=1}^m a_j(x)P_j}{\sum_{j=1}^m a_j(x)} \tag{A.20}$$

if the peaks P_j of the then-part sets B_j equal the then-part set centroids c_j and if all then-part sets B_j have the same nonzero area or volume V_j ; $P_j = c_j$ and $V_1 = \dots = V_m > 0$.

Appendix B. Learning in SAMs: unsupervised clustering and supervised gradient descent

A fuzzy system learns if and only if its rule patches move or change shape in the input–output

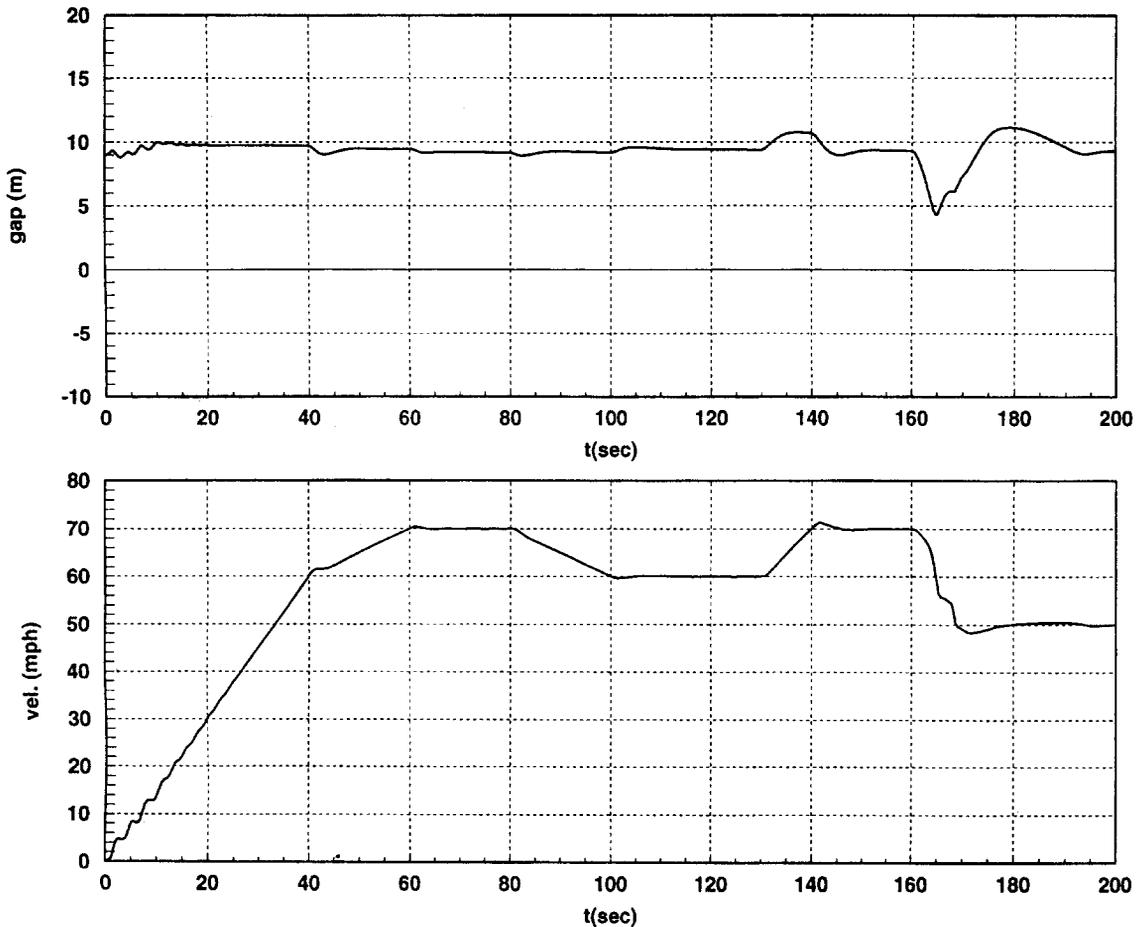


Fig. 35. Simulation results combining the throttle and brake controllers for a platoon deceleration of $0.09g$. The follower car approaches the leader but does not collide with it.

product space $X \times Y$. Learning might change the centers or widths of triangle or trapezoidal sets. These changing sets then change the shape or position of the Cartesian rule patches built out of them. The mean-value theorem and the calculus of variations show [16] that optimal lone rules cover the extrema or bumps of the approximand. Good learning schemes [4, 5] tend to quickly move rule patches to these bumps and then move extra rule patches between them as the rule budget allows. Hybrid schemes use unsupervised clustering to learn the positions and number of the first set of fuzzy rule patches and to initialize the gradient descents of supervised learning (see Fig. 4).

Learning changes system parameters with data. Unsupervised learning amounts to blind clustering in the system product space $X \times Y$ to learn and tune the m fuzzy rules or the sets that compose them. Then k quantization vectors $q_j \in X \times Y$ move in the product space to filter or approximate the stream of incoming data pairs $(x(t), y(t))$ or the concatenated data points $z(t) = [x(t) | y(t)]^T$. The simplest form of such *product space clustering* [14] centers a rule patch at each data point and thus puts $k = m$. In general both the data and the quantizing vectors greatly outnumber the rules and so $k \gg m$.

A natural way to grow and tune rules is to identify a rule patch with the uncertainty ellipsoid

[3, 4] that forms around each quantizing vector q_j from the inverse of its positive definite covariance matrix K_j . Then sparse or noisy data grow a patch larger and thus a less certain rule than does denser or less noisy data. Unsupervised competitive learning [14] can learn these ellipsoidal rules in three steps:

$$\|z(t) - q_j(t)\| = \min(\|z(t) - q_1(t)\|, \dots, \|z(t) - q_k(t)\|), \quad (\text{B.1})$$

$$q_i(t + 1) = \begin{cases} q_j(t) + \mu_t[z(t) - q_j(t)] & \text{if } i = j, \\ q_i(t) & \text{if } i \neq j, \end{cases} \quad (\text{B.2})$$

$$K_i(t + 1) = \begin{cases} K_j(t) + v_i[(z(t) - q_j(t))^T(z(t) - q_j(t) - K_j(t))] & \text{if } i = j, \\ K_i(t) & \text{if } i \neq j \end{cases} \quad (\text{B.3})$$

for the Euclidean norm $\|z\|^2 = z_1^2 + \dots + z_{n+p}^2$. The first step (B.1) is the competitive step. It picks the nearest quantizing vector q_j to the incoming data vector $z(t)$ and ignores the rest. Some schemes may count nearby vectors as lying in the winning subset. We used just one winner per datum. This correlation matching approximates a great deal of the competitive dynamics of nonlinear neural networks. The second step updates the winning quantization or “synaptic” vector and drives it toward the centroid of the sampled data pattern class [13]. The third step updates the covariance matrix of the winning quantization vector. We initialize the quantization vector with sample data ($q_i(0) = z(i)$) to avoid skewed groupings and to initialize the covariance matrix with small positive numbers on its diagonal to keep it positive definite. Projection schemes [3–7] can then convert the ellipsoids into coordinate fuzzy sets. Other schemes can use the unlearned joint set function directly. Supervised learning can also tune the eigenvalue parameters of the rule ellipsoids.

The sequences of learning coefficients $\{\mu_t\}$ and $\{v_t\}$ should decrease slowly [14] in the sense of $\sum_{t=1}^{\infty} \mu_t = \infty$ but not too slowly in the sense of $\sum_{t=1}^{\infty} \mu_t^2 < \infty$. In practice $\mu_t \approx 1/t$. The covariance coefficients obey a like constraint as in our choice of $v_t = 0.2[1 - t/1.2N]$ where N is the total num-

ber of data points. The supervised learning schemes below also use a like sequence $\{\mu_t\}$ of decreasing learning coefficients.

Supervised learning changes SAM parameters with error data. The error at each time t is the desired system output minus the actual SAM output: $e_t = d_t - F(x_t)$. Unsupervised learning uses the blind data point $z(t)$ instead of the desired or labeled value d_t . The teacher or supervisor supervises the learning process by giving the desired value d_t at each training time t . Most supervised learning schemes perform stochastic gradient descent on the squared error and do so through iterated use of the chain rule of differential calculus.

Supervised gradient descent can learn or tune SAM systems [7, 17] by changing the rule weights w_j in (B.4), the then-part volumes V_j , the then-part centroids c_j , or parameters of the if-part set functions a_j . The rule weight w_j enters the ratio form of the general SAM system

$$F(x) = \frac{\sum_{j=1}^m w_j a_j(x) V_j c_j}{\sum_{j=1}^m w_j a_j(x) V_j} \quad (\text{B.4})$$

in the same way as does the then-part volume V_j in the SAM theorem. Both cancel from (A.13) if they have the same value – if $w_1 = \dots = w_m > 0$ or if $V_1 = \dots = V_m > 0$. So both have the same learning law if we replace the nonzero weight w_j with the nonzero volume V_j or V'_j :

$$w_j(t + 1) = w_j(t) - \mu_t \frac{\partial E_t}{\partial w_j} \quad (\text{B.5})$$

$$= w_j(t) - \mu_t \frac{\partial E_t}{\partial F} \frac{\partial F}{\partial w_j} \quad (\text{B.6})$$

$$= w_j(t) + \mu_t e_t \frac{p_j(x_t)}{w_j(t)} [c_j - F(x_t)] \quad (\text{B.7})$$

for instantaneous squared error $E_t = \frac{1}{2}(d_t - F(x_t))^2$ with desired-minus-actual error $e_t = d_t - F(x_t)$. We include the rule weights here for completeness. Our fuzzy systems were unweighted and thus used $w_1 = \dots = w_m > 0$. The volumes then change in the same way if they are independent of the weights (which they may not be in some ellipsoidal learning

schemes):

$$V_j(t + 1) = V_j(t) - \mu_t \frac{\partial E_t}{\partial V_j} \tag{B.8}$$

$$= V_j(t) + \mu_t \varepsilon_t \frac{p_j(x_t)}{V_j(t)} [c_j - F(x_t)]. \tag{B.9}$$

The learning law (B.7) follows since $\partial E_t / \partial w_j = -\varepsilon$ and since

$$\frac{\partial F}{\partial w_j} = \frac{a_j(x) V_j c_j \sum_{i=1}^m w_i a_i(x) V_i - a_j(x) V_j \sum_{i=1}^m w_i a_i(x) V_i c_i}{(\sum_{i=1}^m w_i a_i(x) V_i)^2} \tag{B.10}$$

$$= \frac{w_j a_j(x) V_j}{w_j \sum_{i=1}^m w_i a_i(x) V_i} \left[\frac{c_j \sum_{i=1}^m w_i a_i(x) V_i}{\sum_{i=1}^m w_i a_i(x) V_i} - \frac{\sum_{i=1}^m w_i a_i(x) V_i c_i}{\sum_{i=1}^m w_i a_i(x) V_i} \right] \tag{B.11}$$

$$= \frac{p_j(x)}{w_j} [c_j - F(x)] \tag{B.12}$$

from the SAM theorem.

The centroid c_j in the SAM theorem has the simplest learning law:

$$c_j(t + 1) = c_j(t) - \mu_t \frac{\partial E_t}{\partial F} \frac{\partial F}{\partial c_j} \tag{B.13}$$

$$= c_j(t) + \mu_t \varepsilon_t p_j(x_t). \tag{B.14}$$

So the terms w_j , V_j , and c_j do not change when $p_j \approx 0$ and thus when the j th if-part set barely fires: $a_j(x_t) \approx 0$.

Tuning the if-part sets involves more computation since the update law contains an extra-partial derivative. Suppose the if-part set function a_j is a function of l parameters: $a_j = a_j(m_j^1, \dots, m_j^l)$. Then we can update each parameter with

$$m_j^k(t + 1) = m_j^k(t) - \mu_t \frac{\partial E_t}{\partial F} \frac{\partial F}{\partial a_j} \frac{\partial a_j}{\partial m_j^k} \tag{B.15}$$

$$= m_j^k(t) + \mu_t \varepsilon_t \frac{p_j(x_t)}{a_j(x_t)} [c_j - F(x_t)] \frac{\partial a_j}{\partial m_j^k}. \tag{B.16}$$

The exponential if-part set functions can reduce the learning complexity. They have the

form $a_j = e^{f_j(m_j^1, \dots, m_j^l)}$ and obey $\partial a_j / \partial m_j^k = a_j \partial f_j(m_j^1, \dots, m_j^l) / \partial m_j^k$. Then the parameter update (B.16) simplifies to

$$m_j^k(t + 1) = m_j^k(t) + \mu_t \varepsilon_t p_j(x_t) [c_j - F(x_t)] \frac{\partial f_j}{\partial m_j^k}. \tag{B.17}$$

This can arise for independent exponential or Gaussian sets $a_j(x) = \prod_{i=1}^n \exp\{f_j^i(x_i)\} = \exp\{\sum_{i=1}^n f_j^i(x_i)\} = \exp\{f_j(x)\}$. The exponential set function $a_j(x) = \exp\{\sum_{i=1}^n u_j^i(v_j^i - x_i)\}$ has partial derivatives $\partial f_j / \partial u_j^k = v_j^k - x_k(t)$ and $\partial f_j / \partial v_j^k = u_j^k$.

The Gaussian set function

$$a_j(x) = \exp \left\{ -\frac{1}{2} \sum_{i=1}^n \left(\frac{x_i - m_j^i}{\sigma_j^i} \right)^2 \right\}$$

has mean partial derivative $\partial f_j / \partial m_j^k = (x_k - m_j^k) / (\sigma_j^k)^2$ and variance partial derivative $\partial f_j / \partial \sigma_j^k = -(x_k - m_j^k)^2 / (\sigma_j^k)^3$. Such Gaussian set functions reduce the SAM model to Specht's [20] radial basis function network. We can use the smooth update law (B.17) to update nondifferentiable triangles or trapezoids or other sets by viewing their centers and widths as the Gaussian means and variances.

References

- [1] J.A. Dickerson, H.M. Kim and B. Kosko, Hybrid ellipsoidal learning and fuzzy control for platoons of smart cars, *Proc. 3rd Internat. Conf. on Industrial Fuzzy Control and Intelligent Systems (IFIS'93)* (1993).
- [2] J.A. Dickerson, H.M. Kim and B. Kosko, Fuzzy control for platoons of smart cars, *Proc. 3rd IEEE Internat. Conf. on Fuzzy Systems (IEEE FUZZ-94)* (1994) 1632–1637.
- [3] J.A. Dickerson and B. Kosko, Fuzzy function learning with covariance ellipsoids, *IEEE Internat. Conf. on Neural Networks (IEEE ICNN-93)* (1993) 1162–1167.
- [4] J.A. Dickerson and B. Kosko, Fuzzy function approximation with supervised ellipsoidal learning, *World Congr. on Neural Networks (WCNN '93) II* (1993) 9–13.
- [5] J.A. Dickerson and B. Kosko, Virtual worlds as fuzzy cognitive maps, *Presence* 3 (1994) 173–189.
- [6] J.A. Dickerson and B. Kosko, Ellipsoidal learning and fuzzy throttle control for platoons of smart cars, in: R.R. Yager and L. Zadeh, Eds., *Fuzzy Sets, Neural Networks, and Soft Computing* (Van Nostrand Reinhold, New York, 1994).

- [7] J.A. Dickerson and B. Kosko, Fuzzy function approximation with ellipsoidal rules, *IEEE Trans. Systems Man Cybernet.*, August 1996.
- [8] S.K. Halgamuge and M. Glenser, A hierarchical hybrid fuzzy controller for realtime reverse driving support of vehicles with long trailers, *Proc. 3rd IEEE Internat. Conf. on Fuzzy Systems (IEEE FUZZ-94)* (1994) 1207–1210.
- [9] S.K. Halgamuge and M. Glenser, Application of fuzzy, neural and hybrid methods to an autonomously driven vehicle, *World Congr. on Neural Networks (WCNN '95) III* (1995) 1–6.
- [10] J.K. Hedrick, M. Tomizuka and P. Varaiya, Control issues in automated highway systems, *IEEE Control Systems Mag.* **14**(6) (1994) 21–32.
- [11] A. Hsu, F. Eskafi, S. Sachs and P. Varaiya, The design of platoon maneuver protocols for IVHS, PATH Research Report, UCB-ITS-PRR-91-6 (1991).
- [12] P. Ioannou and T. Xu, Throttle and brake control for vehicle following, *32nd Control Design Conf. (32nd CDC) II* (1993) 1885–1890.
- [13] B. Kosko, Stochastic competitive learning, *IEEE Trans. Neural Networks* **2** (1991) 522–529.
- [14] B. Kosko, *Neural Networks and Fuzzy Systems* (Prentice-Hall, Englewood Cliffs, NJ, 1991).
- [15] B. Kosko, Fuzzy systems as universal approximators, *IEEE Trans. Comput.* **43**(11) (1994) 1329–1333; an early version appears in *Proc. 1st IEEE Internat. Conf. on Fuzzy Systems (IEEE FUZZ-92)* (1992) 1153–1162.
- [16] B. Kosko, Optimal fuzzy rules cover extrema, *Internat. J. Intelligent Systems* **10**(2) (1995) 249–255.
- [17] B. Kosko, Combining fuzzy systems, *Proc. IEEE Internat. Conf. on Fuzzy Systems (IEEE FUZZ-95) IV* (1995) 1855–1863.
- [18] B.C. Kuo, *Automatic Control Systems* (Prentice-Hall, Englewood Cliffs, NJ, 4th ed., 1982).
- [19] S. Sheikholeslam and C.A. Desoer, Longitudinal control of a platoon of vehicles with no communication of lead vehicle information: a system level study, PATH Technical Memorandum, 91-2 (1991).
- [20] D.F. Specht, A general regression neural network, *IEEE Trans. Neural Networks* **2** (1991) 568–576.
- [21] M. Sugeno, An introductory survey of fuzzy control, *Inform. Sci.* **36** (1985) 59–83.