CALIFORNIA PATH PROGRAM
INSTITUTE OF TRANSPORTATION STUDIES
UNIVERSITY OF CALIFORNIA, BERKELEY

Fuzzy Throttle and Brake Control for Platoons of Smart Cars

Hyun Mun Kim
Julie Dickerson
Bark Kosko

University of Southern California

California PATH Research Report
UCB-ITS-PRR-95-42

This work was performed as part of the California PATH Program of the University of California, in cooperation with the State of California Business, Transportation, and Housing Agency, Department of Transportation; and the United States Department of Transportation, Federal Highway Administration.

December 1995

ISSN 1055-1425

To appear in Fuzzy Sets and Systems 95

Fuzzy Throttle and Brake Control for Platoons of Smart Cars

Hyun Mun Kim, Julie Dickerson, [1] and Bart Kosko
Signal and Image Processing Institute
Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, CA 90089-2564

Abstract

Additive fuzzy systems can control the velocity and the gap between cars in single ***lane platoons. The 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 ***decelemtes to slower speeds. We modeled the brake controller using the real test data from the bmke system. A logic switch for throttle and bmke decides*** which ***system to use. The gap controller uses only data 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 down hills and slows. An adaptive throttle controller uses a neuml system to learn the fuzzy rules for different vehicle types.***

Key Words: 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

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

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 I shows: F(z) $= E[Y|X = x]$. The fuzzy platoon controller uses rules that acts 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 x Y: $A \times B \subset$ X x 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 Figure 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 the 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 down hills the controlled car got close to the leader.

We next designed a throttle and brake controller. The combined system let 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

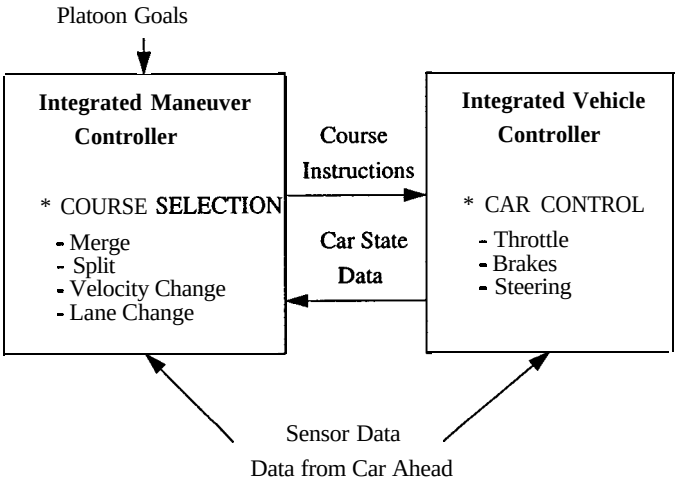

Figure 1: Block diagram of the fuzzy platoon controller.

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 [2] used unsupervised learning to quickly pick the first set of ellipsoidal fuzzy rules. Then supervised learning tunes 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 $F: R^n \to R$ 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 $a_j : R^n \to [0, 1]$ and $b_j : R \to [0, 1]$. 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 $a_j(x) = min(a_j^1(x_1), \ldots, a_j^n(x_n))$ or any other conjunctive form for input vector $x = (x_1, , \ldots, x_n) \in R^n$.

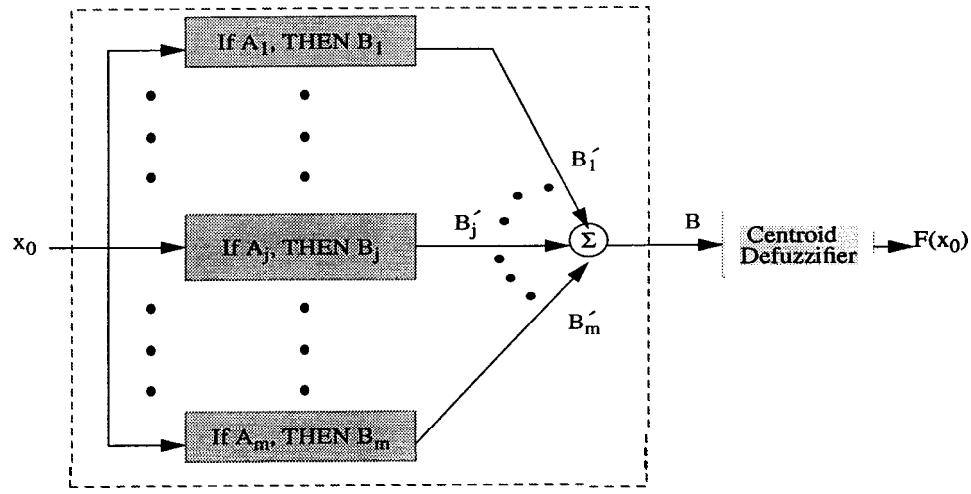

Figure *2:* Architecture of an additive fuzzy system $F: R^n \to R^p$ with m rules. Each input $x_0 \in R^n$ enters the system F as a numerical vector. At the set level x_0 acts as a delta pulse $\delta(x - x_0)$ that combs the if-part fuzzy sets A_j and gives the m set values $a_j(x_0)$: $a_j(x_0) = \int_{R^n} \delta(x - x_0) a_j(x) dx$. 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.

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 "fired" then part sets B_j':

$$B = \sum_{j=1}^{m} B_j' = \sum_{j=1}^{m} a_j(x)\, B_j \qquad (1)$$

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

Figure 3 shows how three rule patches can cover part of the graph of a scalar function $f: R \to R$. The patch cover shows that all fuzzy systems $F: R^n \to R^p$ 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 $f: R^n \to R^p$ where k is the number of sets in each dimension.

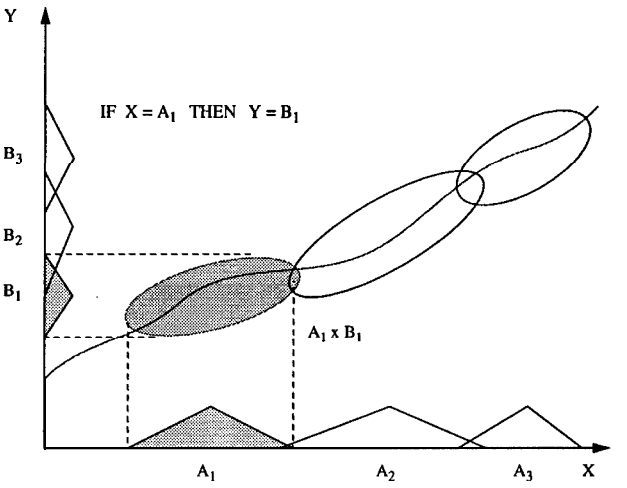

Figure 3: Each fuzzy rule defines a Cartesian-product patch or fuzzy subset of the input-output state space. The fuzzy system approximates a function as it covers its graph with rule patches. Lone optimal rule patches cover extrema.

Optimal rules can help deal with the exponential rule explosion. Lone or local mean-squared optimal rule patches cover the extrema [16] of the approximand 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 I shows that taking the centroid of B in (1) gives [14]-[17] the SAM ratio

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

V_j is the nonzero volume or area of 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 $V_1 = \ldots = V_m > 0$.

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 II uses gradient descent to derivethe supervised learning laws that tune the SAM parameters a_j, V_j, and c_j. The next section shows how to apply

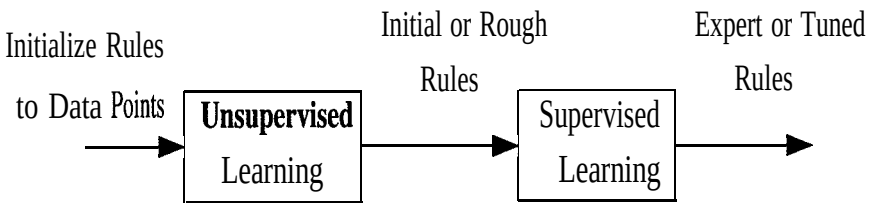

Figure 4: A neural-fuzzy system can learn and tune the fuzzy rules with a hybrid of unsupervised and supervised learning. Unsupervised competitive learning initializes the supervised gradient-descent search for rules that locally minimize the mean-squared error of the function approximation.

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 as shown in Figure 4.

3.1 Unsupervised Rule Estimation with Competitive Learning

A fuzzy rule patch can take the form of an ellipsoid [2]. 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 $q = n + p$ (Figure 5). The ellipsoid is the locus of all z that satisfy [7]

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

where α is a positive real number and m is the center of the ellipsoid in R^q. P has eigenvalues $\lambda_1, \ldots, \lambda_q$. The eigenvalues define the ellipsoid axes. The Euclidean half-lengths of the axes equal $\alpha/\sqrt{\lambda_1}, \ldots, \alpha/\sqrt{\lambda_q}$. 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

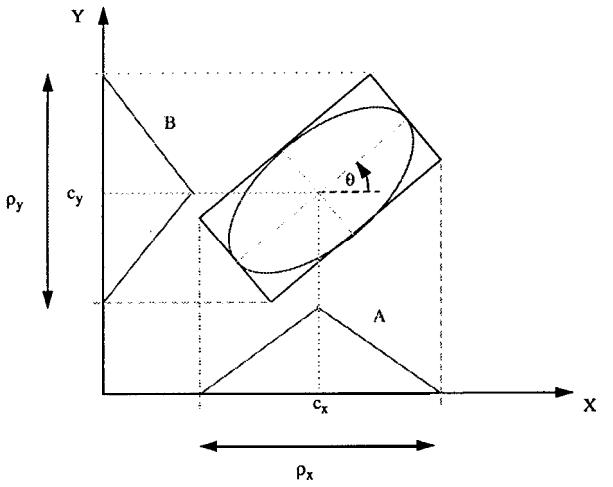

Figure 5: A positive definite matrix defines an ellipsoid about the center m of the ellipsoid. The eigenvalues of the matrix define the length of the axes. The projections of the ellipsoid onto the axes define the input and output fuzzy sets $\boldsymbol{A}$ and B. ρ_x and ρ_y are the lengths of the ellipsoid's projections onto the x and y axes respectively.

each axis of the ellipse. The direction cosine $\cos\gamma_{kij}$ is the angle between the jth eigenvector and the ith axis for the kth ellipsoid. The projection of the kth hyperrectangle onto the ith 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]

$$q_j(t+1) = q_j(t) + \mu_t[z(t) - q_j(t)] \tag{5}$$

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

The coefficients μ_t and v_t must satisfy the convergence conditions of stochastic approximation [14]. In practice $\mu_t \approx \frac{1}{t}$ and $v_t \approx \frac{1}{t}$. Appendix II 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:

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

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 [2].

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 ith triangular output set is V_i:

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

ρ_{iq} is the base of the ith fuzzy rule patch projection on the qth 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 ith fuzzy set:

$$a_j(x) = \min_j (a_j^i(x)). \tag{9}$$

a_j^i is the triangular input fuzzy set for the ith ellipsoid projected on the jth axis:

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

The supervised algorithm uses an iterative form of gradient descent:

$$E_i(k+1) = \mathrm{E;(k)} + \Lambda_{c_k} \Delta_{E_i} SE_k \tag{11}$$

E;(k) is a concatenated vector of the ith ellipsoid's parameters. Λ_{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 ith ellipsoid.

The chain rule of differential calculus gives the supervised ellipsoidal algorithm

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

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

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

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

The partial derivatives [2] in (12)-(15) have the form

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

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

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

$$\frac{\partial a_i^k}{\partial \lambda_{ij}^k} = \begin{cases} -\frac{2\alpha_i |x - c_{x_{il}}|\,|cos\gamma_{ilj}|}{\rho_{il}^2 \sqrt{\lambda_{ij}^3}} & \text{if } |x - c_{x_{il}}| \le \rho_{il}/2 \\ 0 & \text{if } |x - c_{x_{il}}| > \rho_{il}/2 \end{cases} \tag{19}$$

$$\frac{\partial a_i^k}{\partial c_{x_{il}}^k} = \begin{cases} \frac{2}{\rho_{il}} & \text{if } 0 < x - c_{x_{il}} \le \rho_{il}/2 \\ 0 & \text{else} \\ -\frac{2}{\rho_{il}} & \text{if } -\rho_{il}/2 < x - c_{x_{il}} \le 0 \end{cases} \tag{20}$$

$$\frac{\partial V_i^k}{\partial \gamma_{iqj}^k} = \begin{cases} \frac{\alpha_i sin\gamma_{iqj}}{\sqrt{\lambda_{ij}}} & if\ |\gamma_{iqj}| \le \pi/2 \\ \frac{\alpha_i sin\gamma_{iqj}}{\sqrt{\lambda_{ij}}} & \text{if } |\gamma_{iqj}| > \pi/2 \end{cases} \tag{21}$$

$$\frac{\partial m_i^k}{\partial \gamma_{ilj}^k} = \begin{cases} \frac{4\alpha_i |x - c_{x_{il}}| sin\gamma_{ilj}}{\rho_{il}^2 \sqrt{\lambda_{ij}}} & \text{if } |x - c_{x_{il}}| \le \rho_{il}/2 \text{ and } |\gamma_{ilj}| \le \pi/2 \\ -\frac{4\alpha_i |x - c_{x_{il}}| sin\gamma_{ilj}}{\rho_{il}^2 \sqrt{\lambda_{ij}}} & \text{if } |x - c_{x_{il}}| \le \rho_{il}/2 \text{ and } |\gamma_{ilj}| \le \pi/2 \\ 0 & \text{if } |x - c_{x_{il}}| > \rho_{il}/2 \end{cases} \tag{22}$$

$$\frac{\partial V_i^k}{\partial \lambda_{ij}^k} = -\frac{\alpha_i |cos\gamma_{iqj}|}{2\sqrt{\lambda_{ij}^3}} \tag{23}$$

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 ith car in the platoon has two inputs:

$$\Delta v_i(t) = v_{platoon} - v_i(t) \tag{24}$$

$$a_i(t) = a_i(t). \tag{25}$$

The output is the change in throttle angle $\partial_{throttle}$. 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 ($\partial_{throttle}$ is medium negative(MN)). The velocity difference and the acceleration each have 7 fuzzy if-part sets. The number of fuzzy rules for the velocity controller is $7 \times 7 = 49$. Figures 6-7 show the fuzzy sets and rules for this controller [4].

4.2 Gap Controller

Figure 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 Ad;(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 ith car are:

$$\Delta d_i(k) = d_{desired} - d_i(k) \tag{26}$$

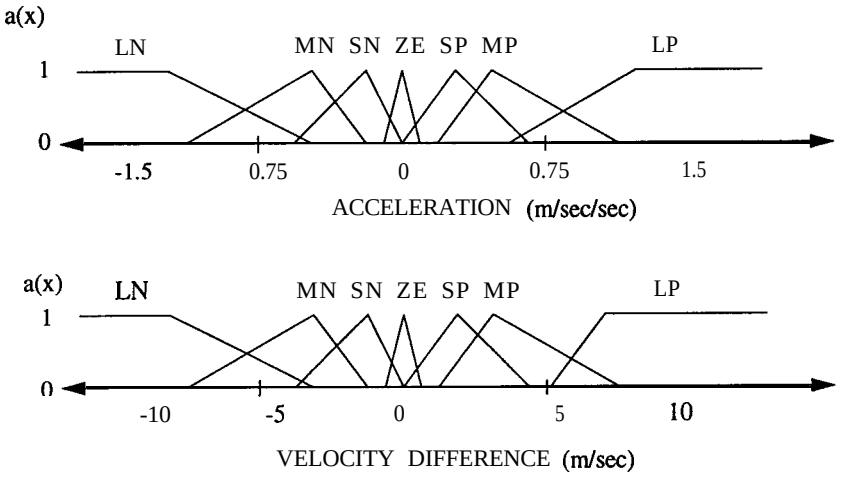

Figure 6: Fuzzy sets for velocity controller.

VELOCITY DIFFERENCE

ACCELERATION	LP	MP	SP	ZE	SN	MN	LN
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

Figure 7: Fuzzy rules for velocity controller.

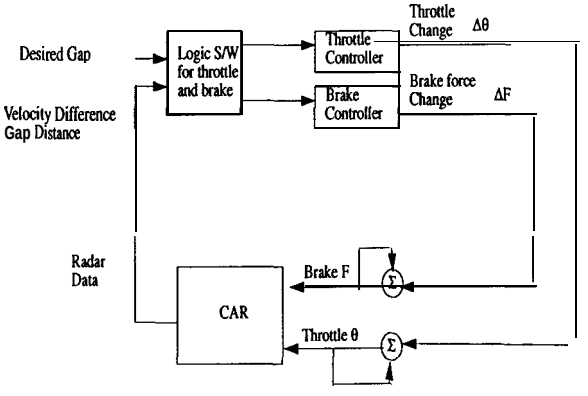

Figure 8: Block diagram of the fuzzy gap controller that uses the brake and throttle.

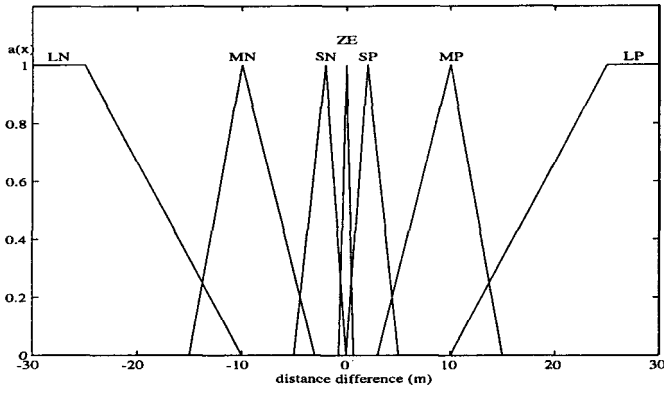

Figure 9: If-part fuzzy set functions for distance difference input for the gap controller. The narrower sets near *ZERO* give finer control near the desired or equilibrium position.

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

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

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 MN) AND (Ad;(k) is ZE) THEN ($\partial_{throttle}$ is MP). Figures 9 through 11 show the fuzzy set values for the gap controller fuzzy variables $\Delta d_i, \Delta v_i$, and Δa_i.

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

The gap controller had only 3 fuzzy sets of acceleration difference as shown in Figure 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 input by using the estimated

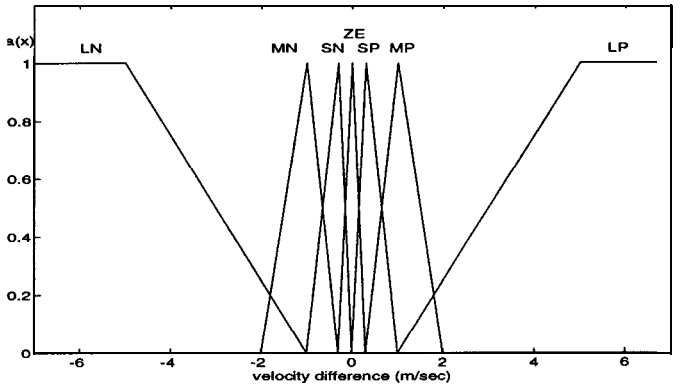

Figure 10: If-part fuzzy set functions for velocity difference input for the gap controller.

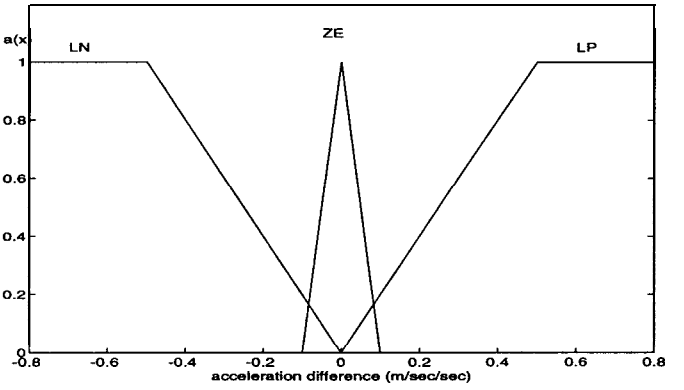

Figure 11: If-part fuzzy set functions for acceleration difference input for the gap controller.

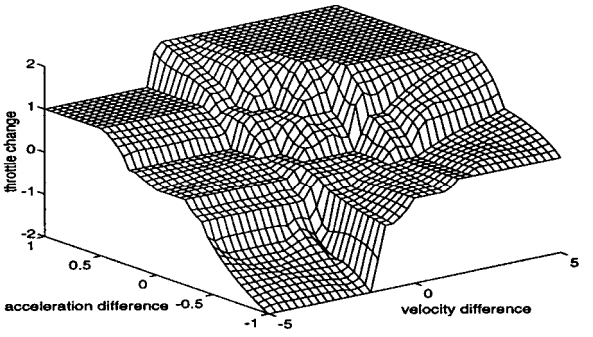

Figure 12: Gap control surface when distance difference is large negative (LN). Acceleration difference is m/sec^2, velocity diference is m/sec, and throttle change unit is 0.02 degree.

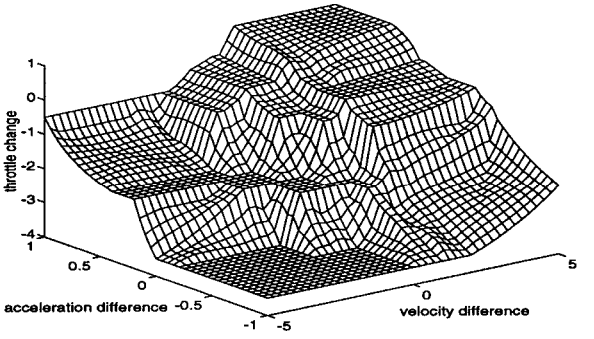

Figure 13: Gap control surface when distance difference is *large positive* (LP).

acceleration difference as described below. Figures 12 and 13 show the control surfaces for different values of the distance error Ad;.

The throttle actuator had a mechanical delay of 0.25 seconds. 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 acceleration difference change greatly. It also generates high frequency terms of throttle change.

We instead approximate the acceleration input in (28) as the difference of the velocity measure-

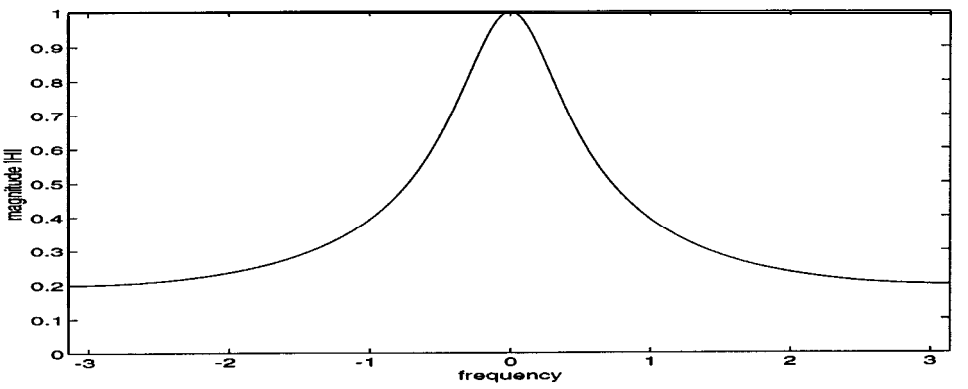

Figure 14: Frequency response of our filter $H_{LP}(z)$.

ments:

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

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 seconds for the sampling time and c = 20 x 0.03048 = 0.6096.

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

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

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

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

smoothes $\theta_i(k)$. 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. Figure 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 $1/3$ to decrease round-off errors.

Figure 15: Block diagram of brake controller.

4.3 Brake Controller

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

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

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

Figures 16 and 17 show the fuzzy set values for the brake controller fuzzy variables Δd_i and Δv_i. One brake rule is If Ad is Medium Positive (MP) and Δv is Medium Negative (MN) then the change in brake actuator is Medium Small (MS). Figures 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(s).

The brake controller has 5 x 5 = 25 fuzzy rules. Figure 19 shows the fuzzy rules for the brake controller. Nine fuzzy sets quantize the fuzzy variables Ad; and Δv_i.

EP: Extreme Positive

LP: Large Positive

MP: Medium Positive

SP: Small Positive

ZE: Zero

SN: Small Negative

MN: Medium Negative

LN: Large Negative

EN: Extreme Negative

Figure 20 shows the control surface of the brake controller. The brakes are on only when the distance

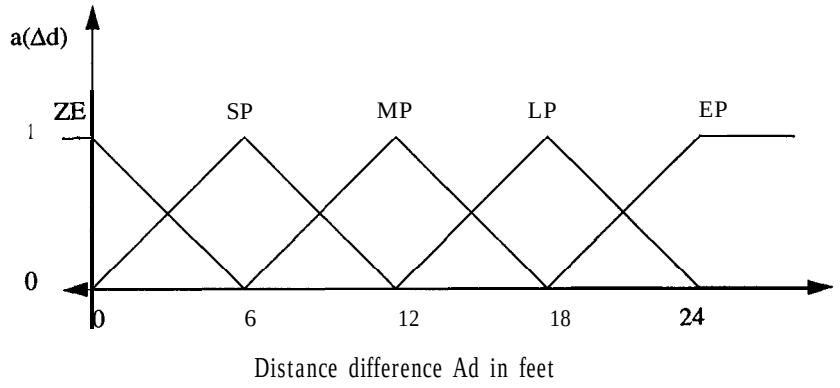

Figure 16: If-part fuzzy set functions for distance difference input to the brake controller.

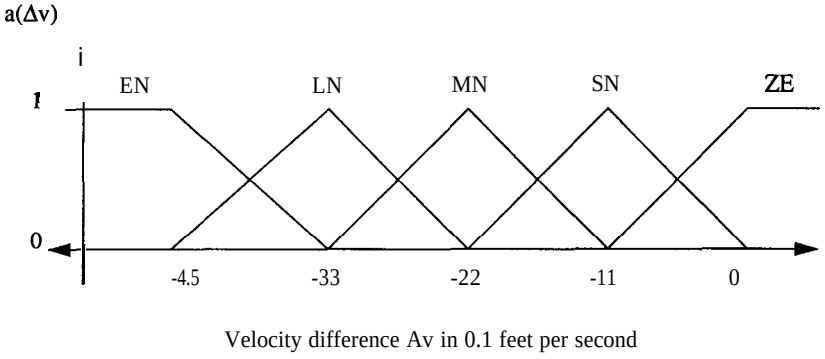

Figure 17: If-part fuzzy set functions for velocity difference input to the brake controller.

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. Figure 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

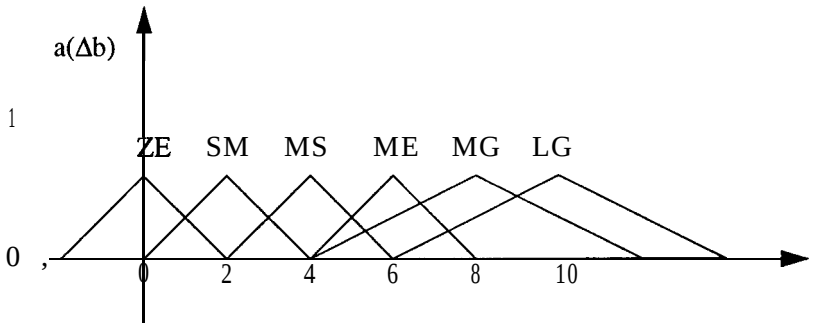

Figure 18: Then-part fuzzy set functions for the brake actuator change. The fuzzy system F uses just the centroids c_j and the areas or volumes V_j of the 6 then-part sets to compute the output F(a). Appendix I gives the details of this computation.

VELOCITY DIFFERENCE

DISTANCE DIFFERENCE	ZE	SN	MN	LN	EN
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

Figure 19: Fuzzy rules for the brake controller. The if-part fuzzy sets are Δv and Ad. The then-part sets give the change Ab in the brake actuator signal.

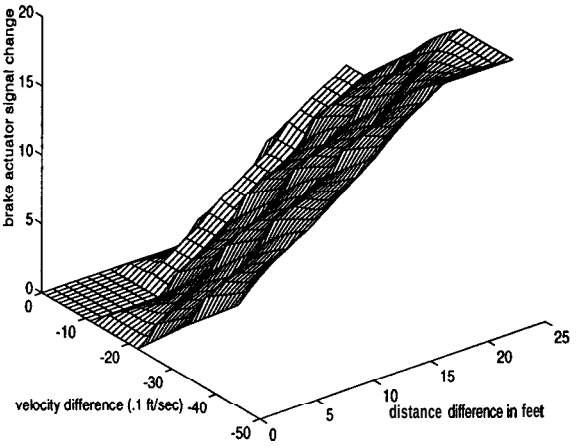

Figure 20: Control surface for fuzzy brake controller. The output unit is 0.15 degree change for brake pedal.

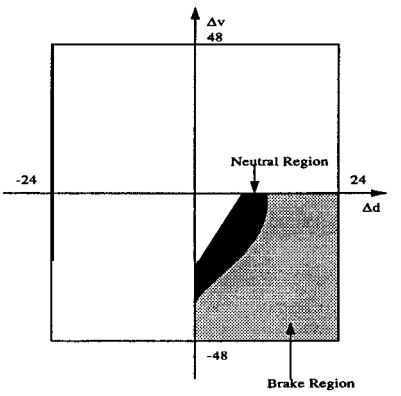

Figure 21: Domain of look-up table for the fuzzy gap controller.

the neutral region. Figure 21 shows the neutral region for the fuzzy brake controller. The neutral region covers small values of the inputs Δv and Ad. The neutral region is that part of the control surface in Figure 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 fuzzy sets and rules for throttle and brake control. Figure 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 $\frac{1}{10} ft/sec$. 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 $(\partial_{brake}$ is MS).

The look-up table for the throttle system had 3 x 49 x 97 = 14,259 entries. The distance difference varied from -24 *ft* to 24 *ft* in 1 foot units and has 49 entries. The velocity difference had 97 entries as it varied from -4.8 ft/sec to 4.8 ft/sec in $1/10\, ft/sec$ units. The look-up table for the brake system had 25 x 49 = 1225 entries since it applied only in the 4th 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" longitudinal 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:

$$m_i a_i = m_i \xi_i - K_{d_i} v_i^2 - d_{m_i} - m_i g \sin \beta_i \tag{36}$$

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

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; and v_i stand for acceleration and

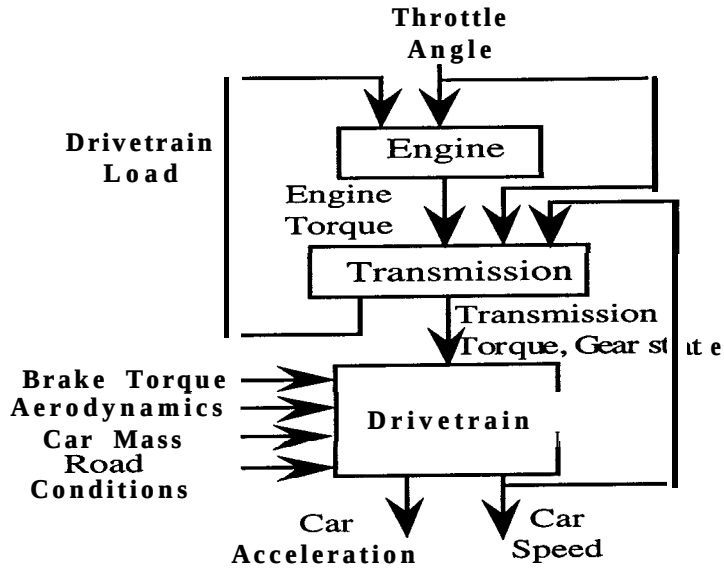

Figure 22: Validated car model from the Ford Motor Company.

velocity. $K_{d_i} v_i^2$ and d_{m_i} 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 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 [4] τ_i = 0.2, $K_{d_i} = 0.44$, and $d_{m_i} = 352$.

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

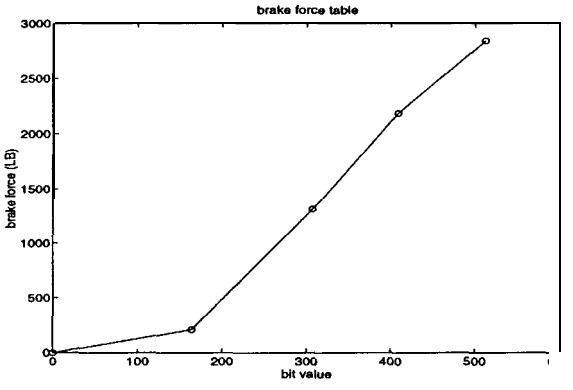

Figure 23: Brake force as a function of the actuator bit value.

5.2 Brake Models

We modeled the brake to test the fuzzy brake controller. The Ford-tuned car model in Figure 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. This 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 pounds.

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

$$\mathrm{Y} = 50 + 0.819 \times V + .0192 \times V^2 \tag{38}$$

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:

$$F_b = F - Y - D \tag{39}$$

F_b is the brake force and D is the force from an external disturbance such as hill. We interpolated between data points to get the curve in Figure 23. It shows the brake force with respect to the actuator bit value using F = mu 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 [4] gave 7,500 training samples in 200 trajectories for a sport utility car. The training vectors (a, $\Delta v, \partial_{throttle}$) 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 II. 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 cell defined the rules [14]. Appendix II 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. Figure 24 compares the platoon velocity for the lone unsupervised and hybrid controllers. The desired velocity was 25 m/sec. The hybrid controller accelerated faster than did 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 seconds. We used the validated car model Figure 22 for these simulations. Figures 25 and 26 show the results for a three-car platoon that changes velocity due to terrain changes such as hills. The desired gap distance was 9 meters. The follower cars maintained the desired gap distance except for the transients at the start of the simulation.

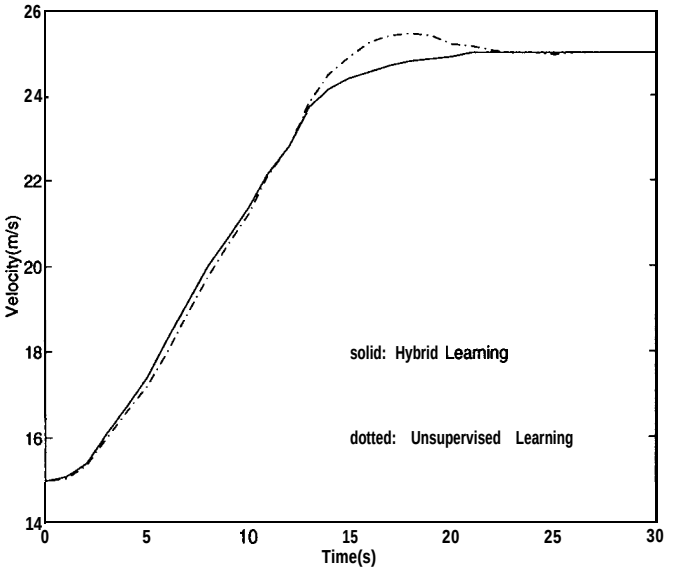

Figure 24: Comparison of the velocity controller performance after lone unsupervised learning and after hybrid learning with unsupervised clustering and supervised gradient descent. The hybrid controller gave a faster response and had no overshoot.

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

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

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. Figures 27 and 28 show that the fuzzy throttle system gave robust control in the presence of noise. The noise had little effect on the velocity of the follower car in the 2-car platoon. The noise did make the gap jiggle but never by more than 1 meter.

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

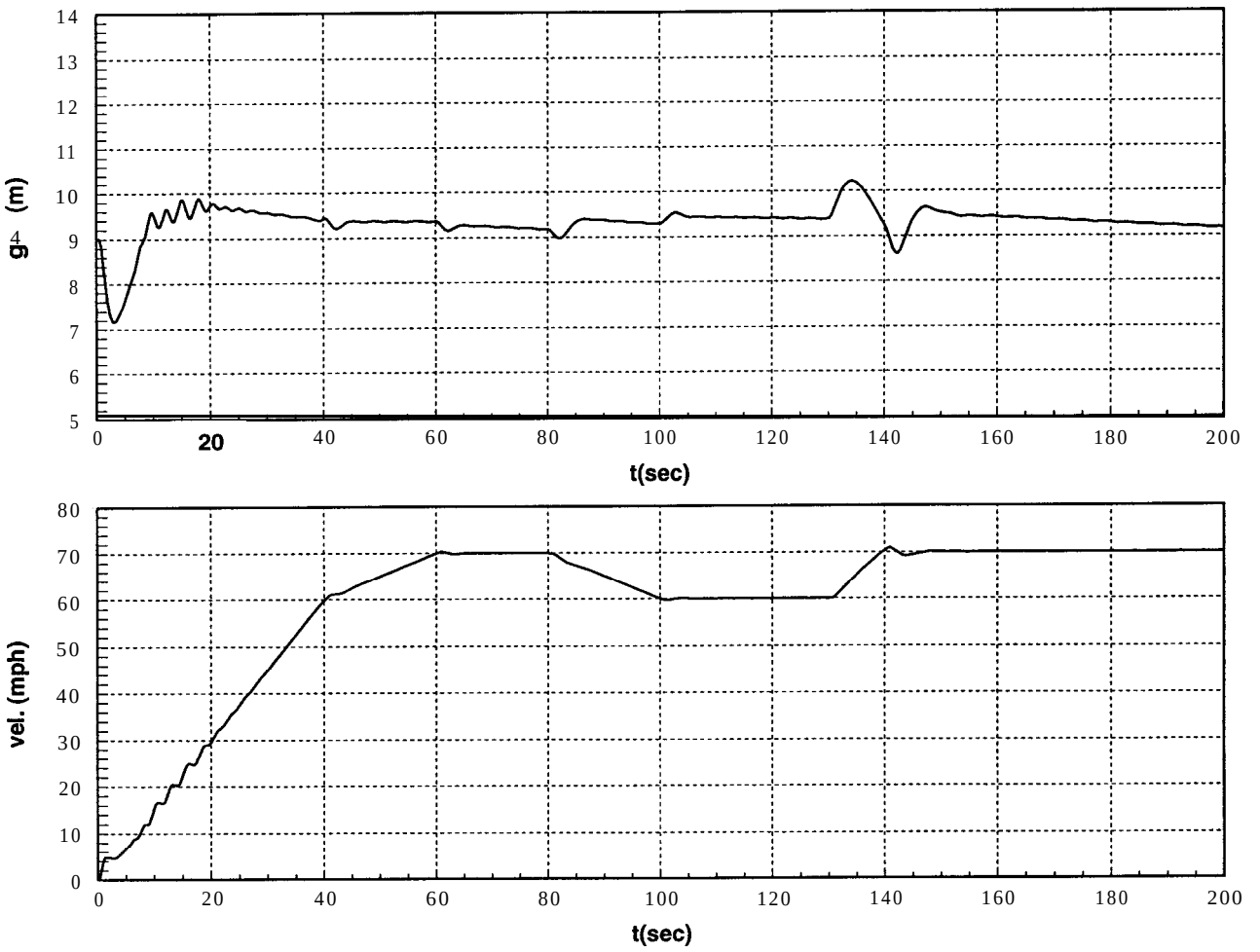

Figure 25: Simulation results for the second follower car in a 3-car platoon. The car changes speed in the presence of hills. It tends to maintain the desired gap of 9 meters between it and the platoon leader. The throttle actuator had a mechanical delay of 0.25 seconds.

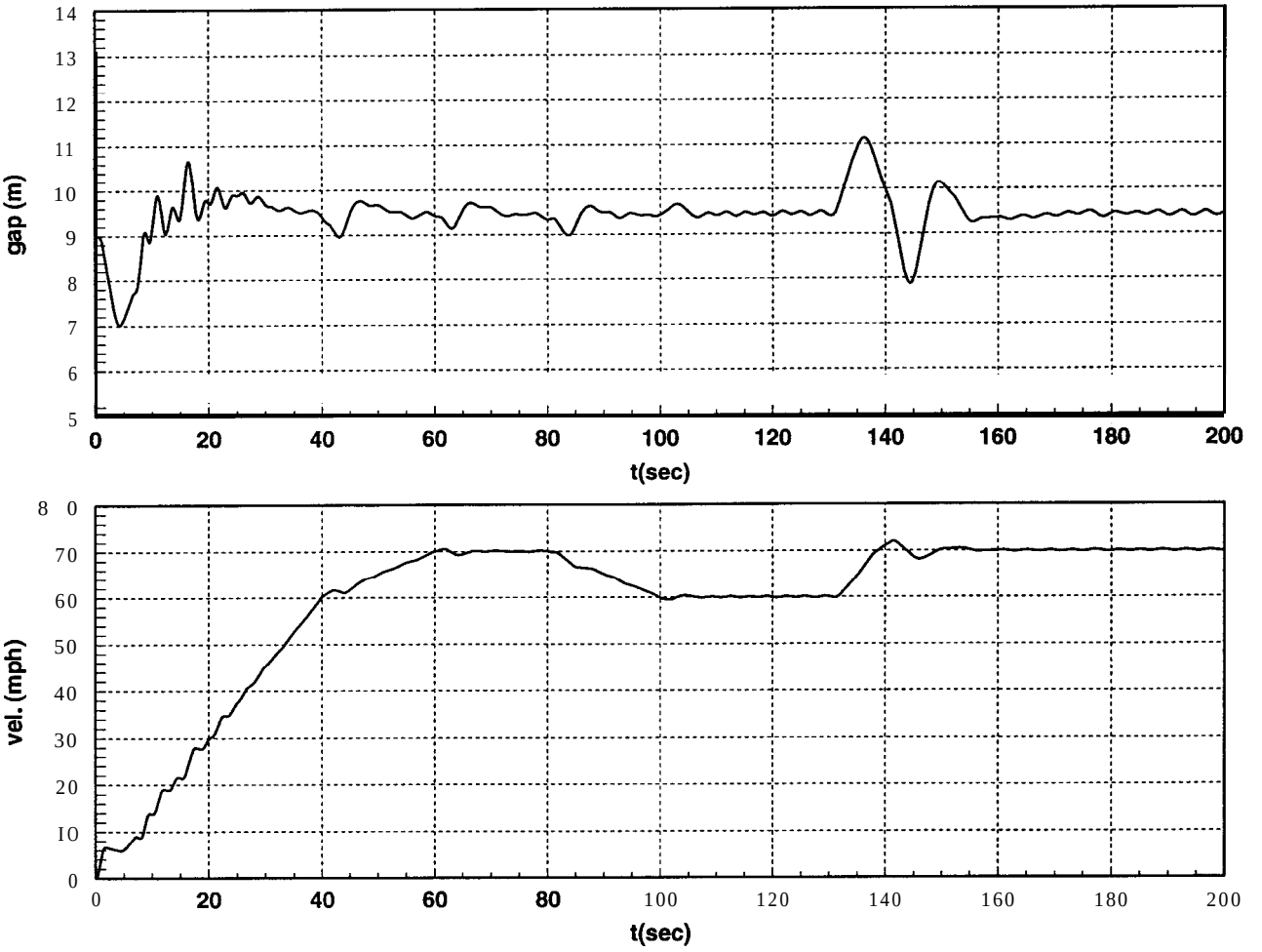

Figure 26: Simulation results for the third follower car in a 3-car platoon. The third car also tends to maintain the desired gap of 9 meters between it and the second car but it does so with more variability.

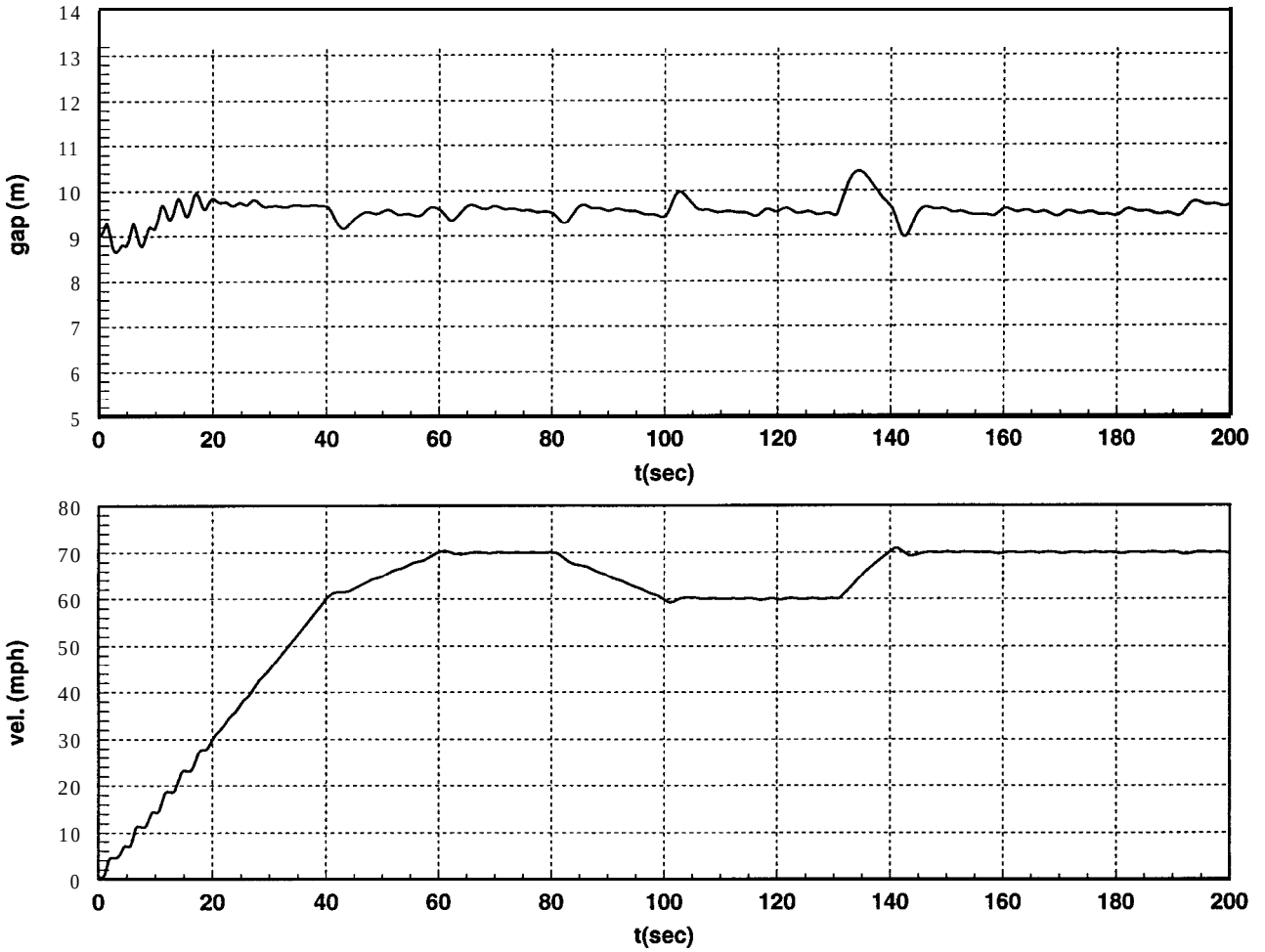

Figure 27: Simulation results for the follower car in a 2-car platoon when the signal-to-noise ratio is *27db*. The gap error due to noise never exceeds 1 meter.

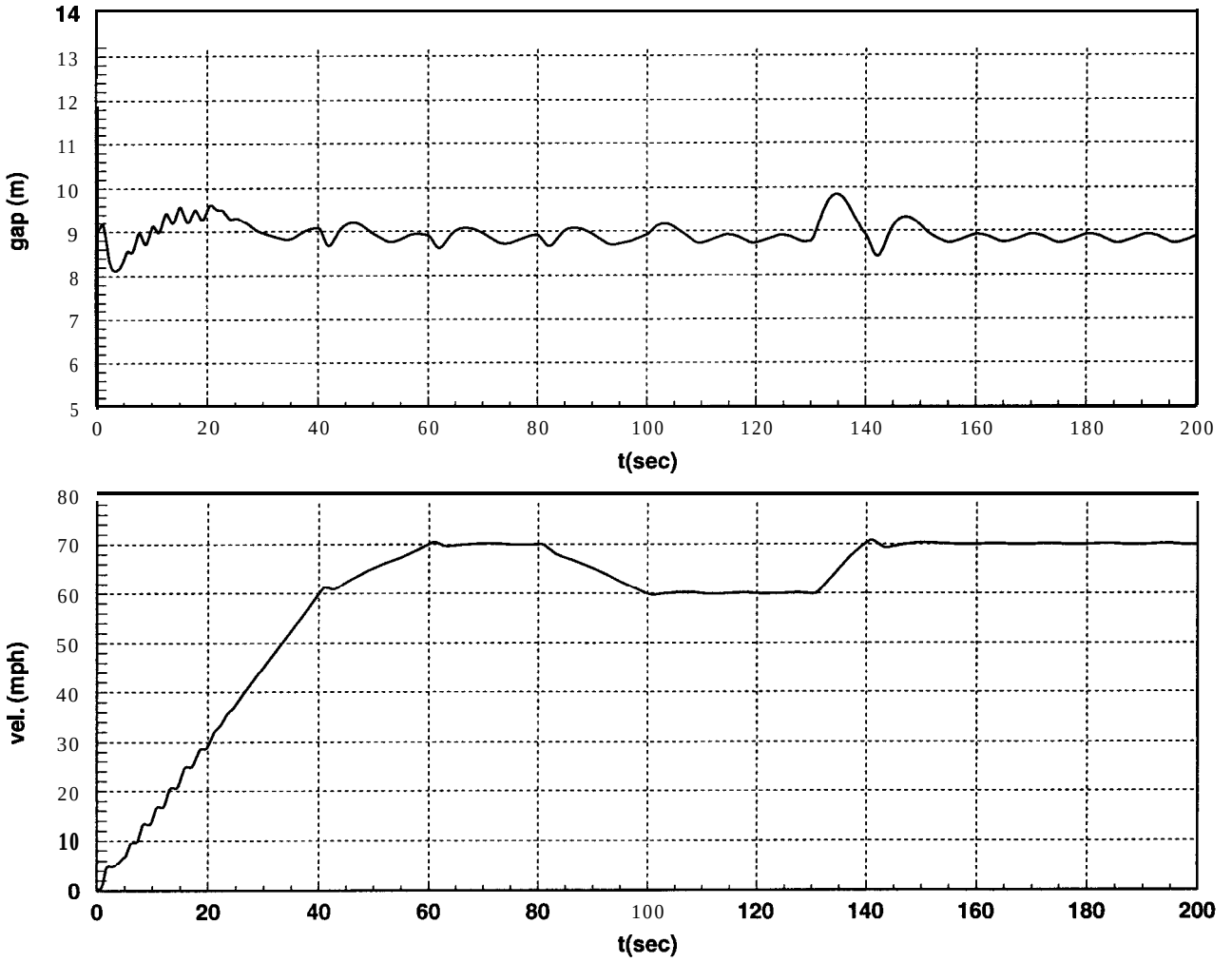

Figure 28: Simulation results for the follower car in a 2-car platoon when the signal-to-noise ratio is lldb. The additive noise has less effect on the car's velocity than on the gap it tries to maintain with the lead car.

car and the car ahead. The radar tracked the car ahead and had a measurement delay of 0.05 seconds. Figure 29 (a) shows the follower car gap as the platoon accelerated. Figure 29 (b) shows the closing rate between the cars. Figure 29 (c) shows the throttle value as the car accelerated. The desired gap was 125 feet. 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 feet. The follower car dropped back as the platoon started up the hill. Figure 30 (a) shows the gap distance as the platoon went up a hill. The follower dropped back and then moved to the right gap. Figure 30 (b) shows the closing rate between the cars. The spike at 15 seconds occurred when the radar sensor briefly lost the lead car in the platoon. The follower car maintained a constant throttle until the sensor detected a new target. Figure 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. Figure 31 shows the leader car's velocity profile and the external disturbances. These disturbances correspond to a 5% grade both up hill and down hill. Figure 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.

Figure 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 meters before the car slows to the desired speed of 60 mph.

The follower cars also need brake control when the platoon slows down. Figure 34 shows the simulation results without using a brake when the platoon decelerates. The leader car decelerates from 70 *mph* to 50 *mph* starting at t = 160s. The leader car maintains 50 *mph* at t = 170s. The deceleration at t = 160s forced the follower car to brake to avoid a collision. Engine torque cannot slow the car down enough. Figure 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.

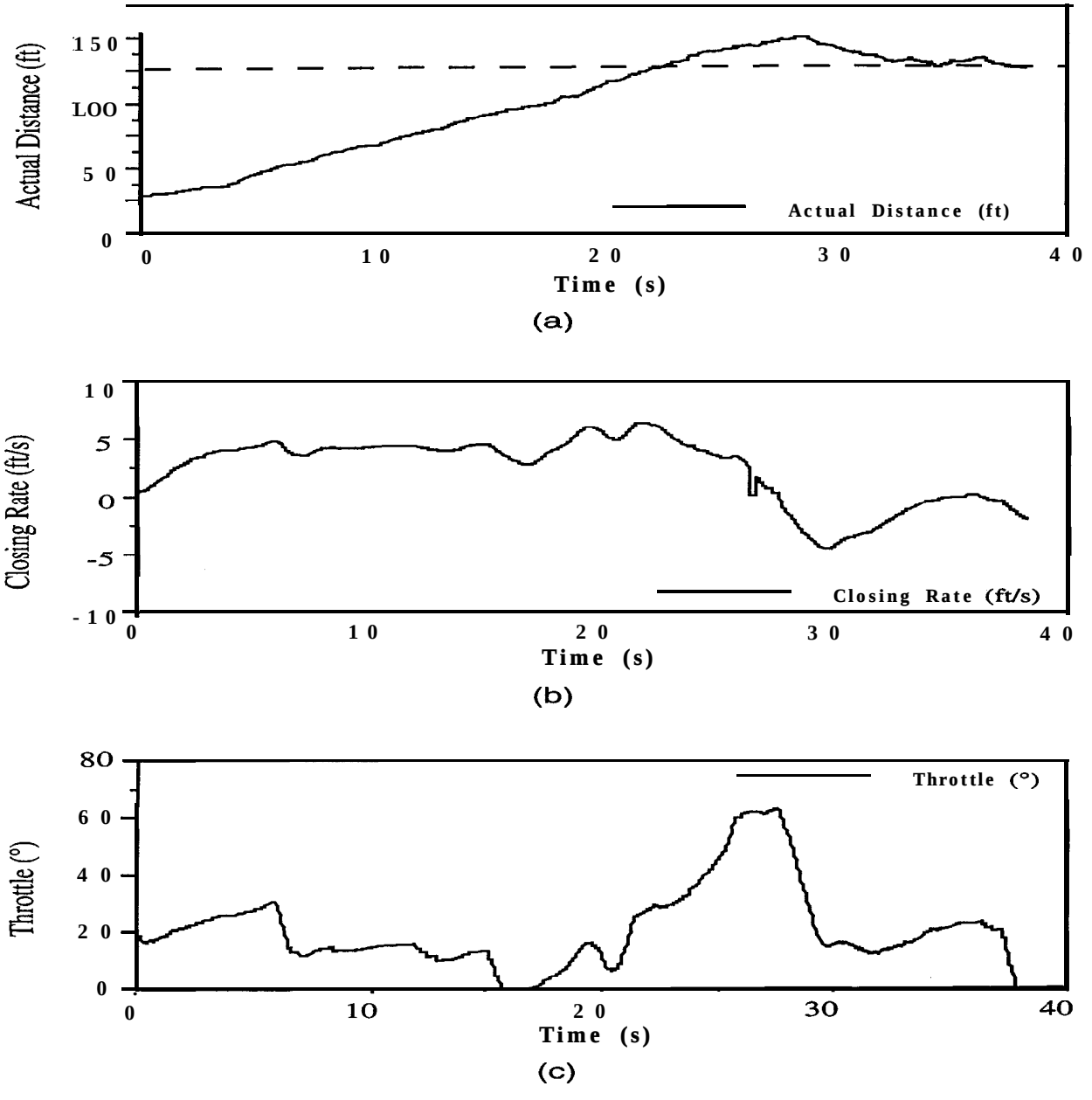

Figure 29: Follower car data as a two-car platoon accelerates on the highway. (a) shows the gap. (b) shows the closing rate between the cars. (c) shows the throttle values for the follower cars. The follower car drops back because the initial gap was less than the desired gap of 125 feet. The radar had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

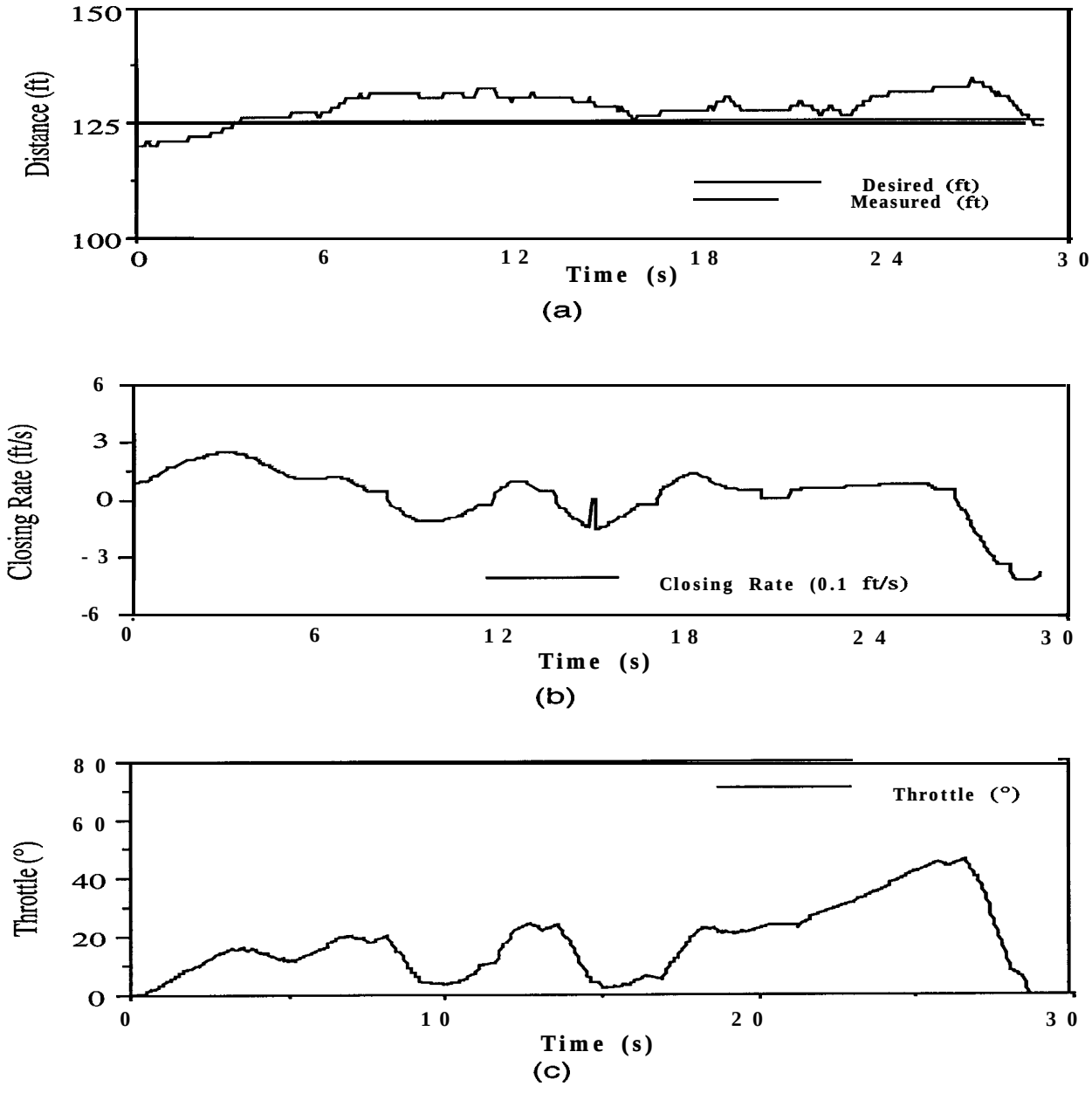

Figure 30: Follower car data as a two-car platoon climbs a hill. (a) shows the gap. (b) shows the closing rate between the cars. The spike at 15 seconds shows where the radar sensor briefly lost the lead car in the platoon. (c) shows the throttle values for the follower cars as the platoon went uphill.

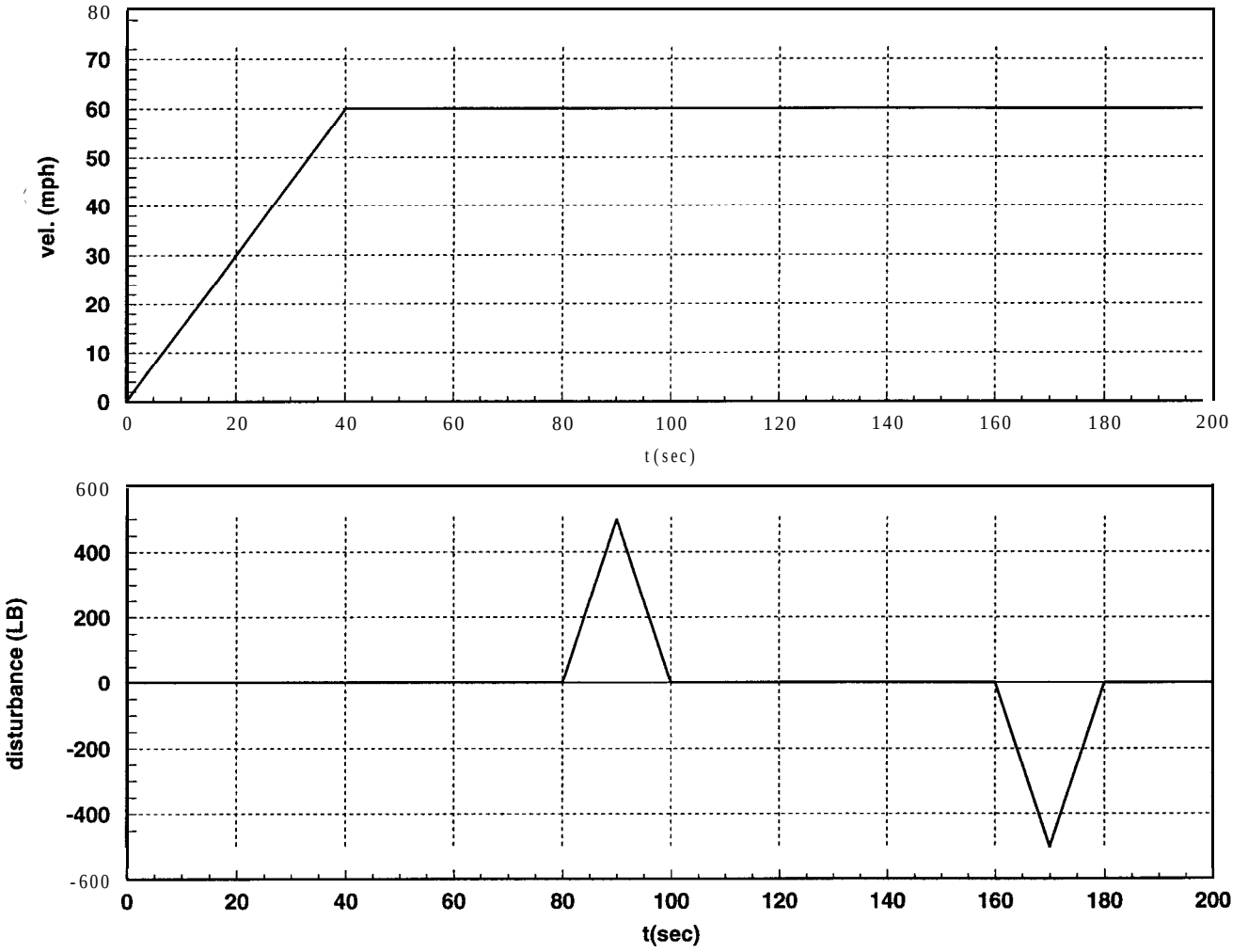

Figure 31: Profile of leader car in a 2-car platoon. External forces that can model hills. The triangles model both a 5% grade uphill and downhill.

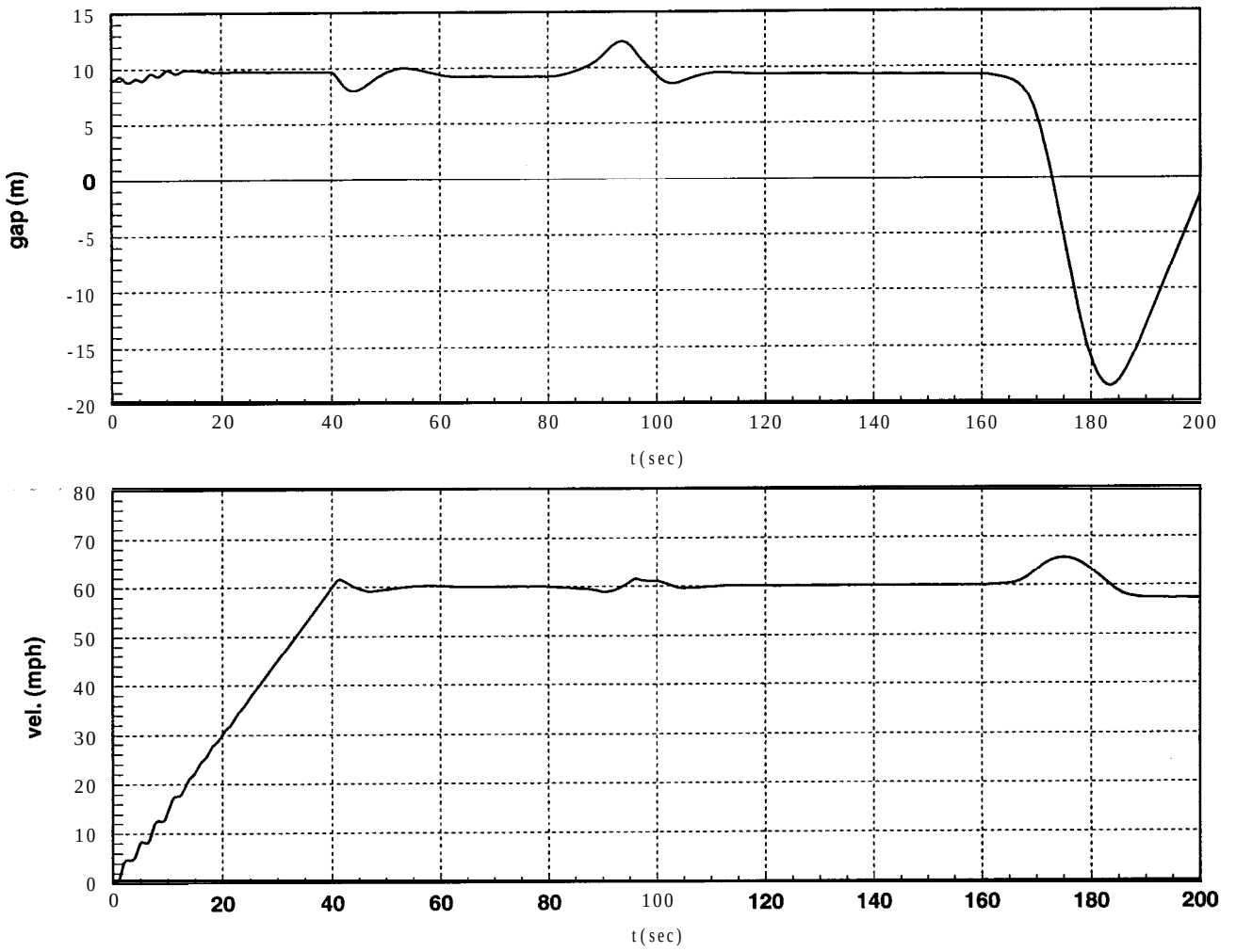

Figure 32: Simulation results for the follower car using throttle only for up and down hills. The follower car cannot slow down enough to avoid hitting the lead car on the downhill grade ($t \approx 170$s).

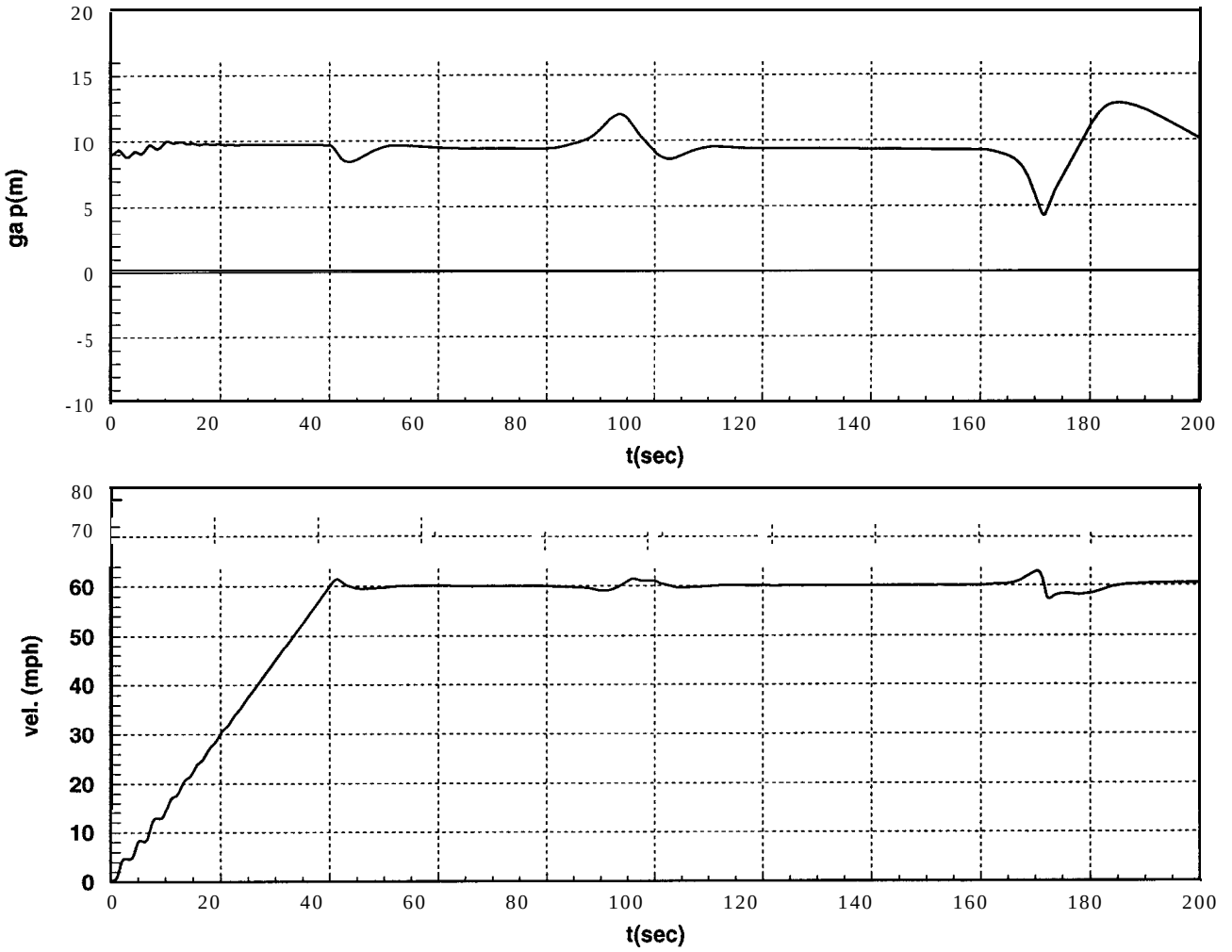

Figure 33: Simulation results for the follower car with a gap controller that combines throttle and brakes. The cars follow the velocity and terrain profile in Figure 24. The follower car approaches the leader but does not collide with it.

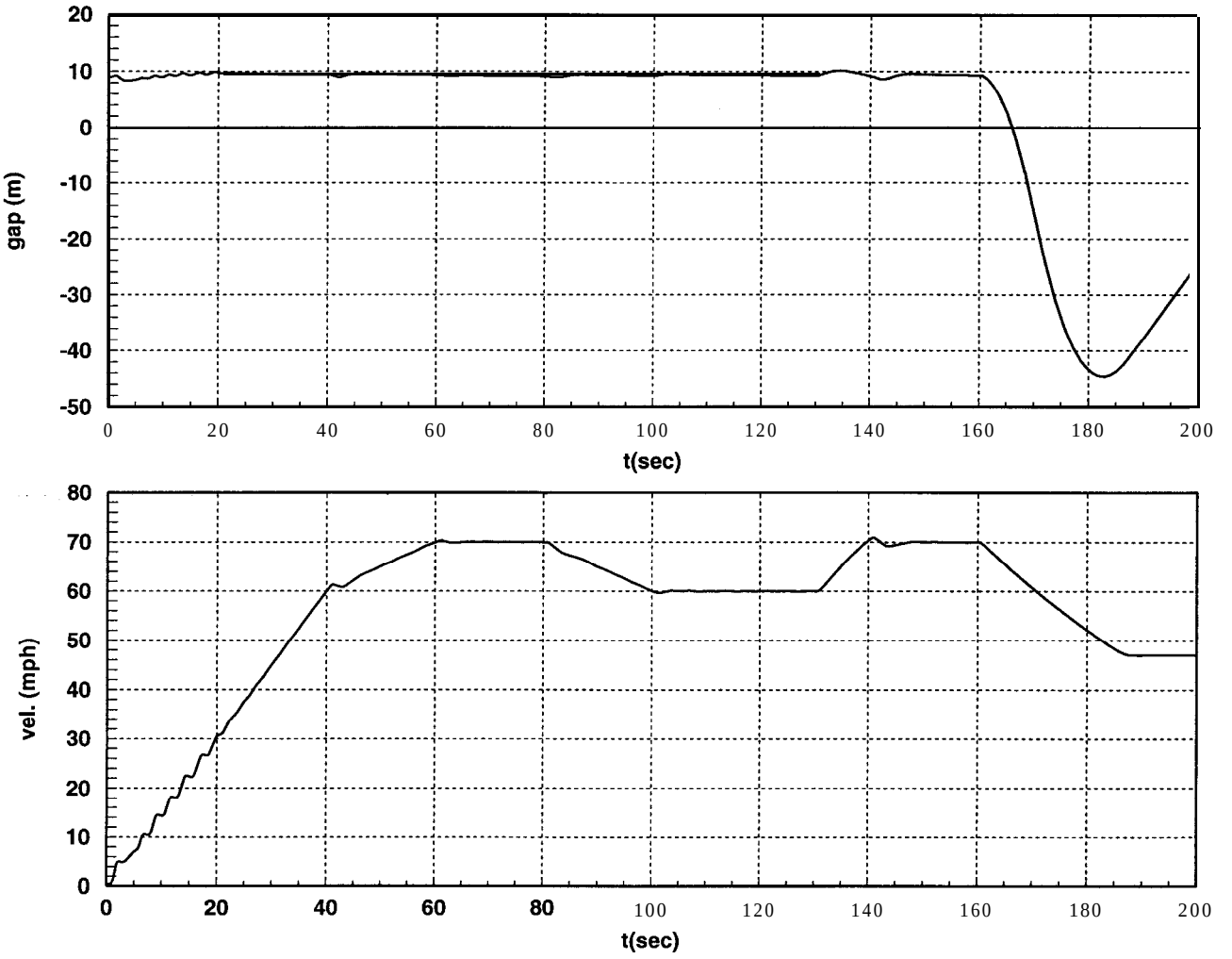

Figure 34: Simulation results using throttle only for a lead car deceleration of $0.09g$. The follower car collides with the leader at $t \approx 165$s.

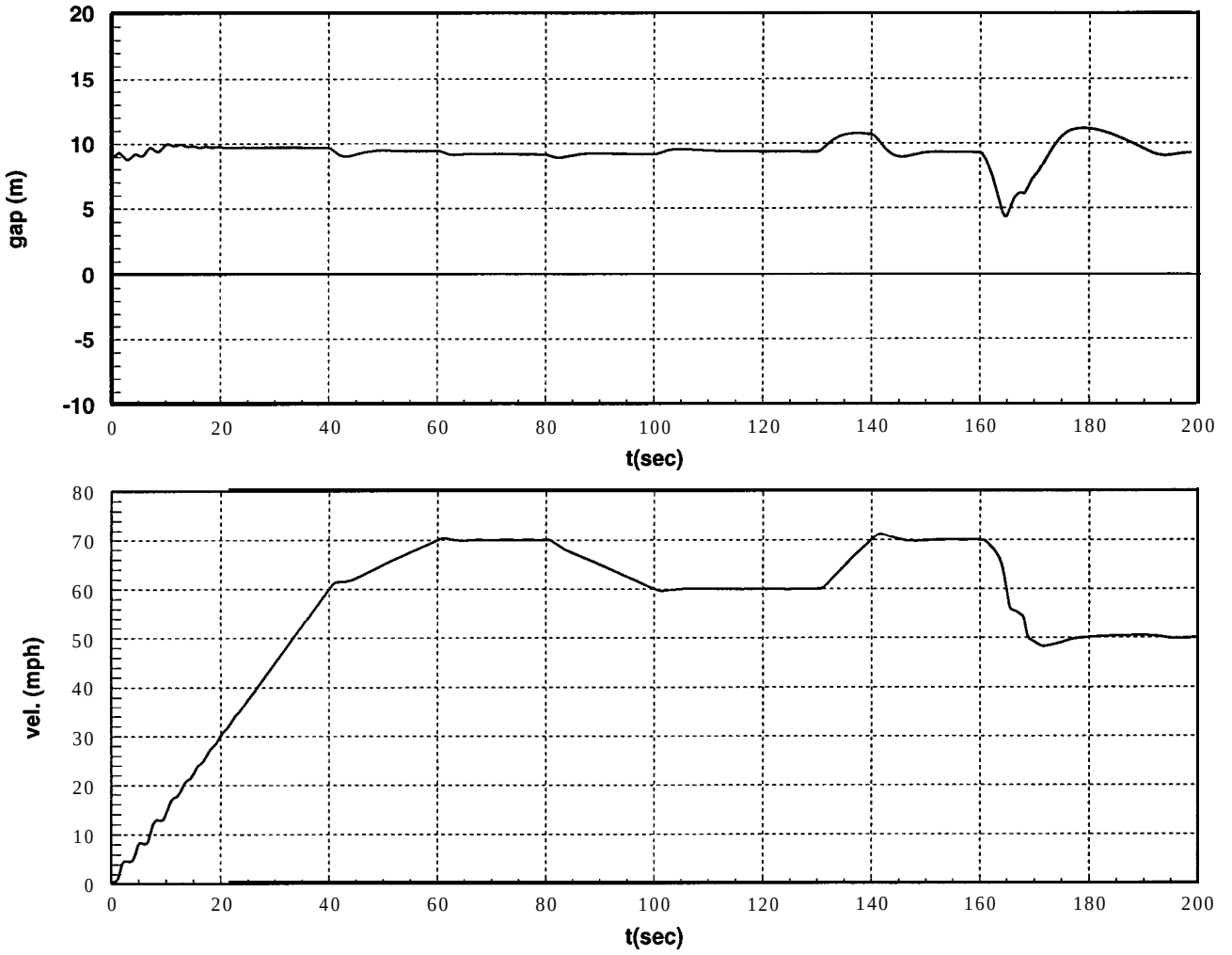

Figure 35: Simulation results combining throttle and brake controller for a platoon deceleration of $0.09g$. The follower car approaches the leader but does not collide.

7 Conclusion

An additive fuzzy system 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 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. Online 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.

Acknowledgment

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.

References

[1] Dickerson, J. A., and Kosko, B., "Fuzzy Function Learning with Covariance Ellipsoids," *IEEE International Conference on Neural Networks (IEEE ICNN-93)*, 1162-1167, March 1993.

[2] Dickerson, J. A., and Kosko, B., "Fuzzy Function Approximation with Supervised Ellipsoidal Learning," World Congress on Neural *Networks (WCNN'93)*, Vol. II, 9-13, July 1993.

[3] Dickerson, J. A., and Kosko, B., "Virtual Worlds as Fuzzy Cognitive Maps," Presence, Vol. 3, No. 2, 173-189, Spring 1994.

[4] Dickerson, J. A., and Kosko, B., "Ellipsoidal Learning and Fuzzy Throttle Control for Platoons of Smart Cars," in *Fuzzy Sets,* Neural *Networks, and* Soft Computing, R. R. Yager, L. Zadeh, Editors, Van Nostrand Reinhold, 1994.

[5] Dickerson, J. A., and Kosko, B., "Fuzzy Function Approximation with Ellipsoidal Rules," *IEEE Transactions on* Systems, Man, and Cybernetics, to appear, 1995.

[6] Dickerson, J. A., Kim, H. M. and Kosko, B., "Hybrid Ellipsoidal Learning and Fuzzy Control for Platoons of Smart Cars," in the Proceedings *of the 3rd International* Conference *on* Industrial *Fuzzy* Control *and Intelligent Systems (IFIS'93)*, December, 1993.

[7] Dickerson, J. A., Kim, H. M. and Kosko, B., "Fuzzy Control for Platoons of Smart Cars," *Proceedings of the 3rd IEEE* International Conference *on Fuzzy Systems (IEEE FUZZ-94)*, 1632-1637, July 1994.

[8] Halgamuge, S. K., and Glenser, M., "A Hierarchical Hybrid Fuzzy Controller for Realtime Reverse Driving Support of Vehicles with Long Trailers," *Proceedings of the 3rd IEEE International* Conference *on Fuzzy Systems (IEEE FUZZ-94)*, 1207-1210, July 1994.

[9] Halgamuge, S. K., and Glenser, M., "Application of Fuzzy, Neural and Hybrid Methods to an Autonomously Driven Vehicle," *World Congress* on *Neural Networks (WCNN'95)*, Vol. III, 1-6, July 1995.

[10] Hedrick, J. K., Tomizuka, M. and Varaiya, P., "Control Issues in Automated Highway Systems," *IEEE Control Systems Magazine,* Vol. 14, No. 6, 21-32, December 1994.

[11] Hsu, A., Eskafi, F., -Sachs, S., and Varaiya, P., "The Design of Platoon Maneuver Protocols for IVHS," *PATH Research Report,* UCB-ITS-PRR-91-6, April 20, 1991.

[12] Ioannou, P., and Xu, T., "Throttle and Brake Control for Vehicle Following," *32nd Control Design* Conference *(32nd CDC),* Vol. II, 1885-1890, December 1993.

[13] Kosko, B., "Stochastic Competitive Learning," *IEEE Transactions on Neural Networks,* Vol. 2, No. 5, 522-529, September 1991.

[14] Kosko, B., Neural *Networks* and *Fuzzy Systems,* Prentice Hall, Englewood Cliffs, NJ, 1991.

[15] Kosko, B., "Fuzzy Systems as Universal Approximators," *IEEE Transactions on Computers,* Vol. 43, No. 11, 1329-1333, November 1994; an early version appears in *Proceedings of the 1st IEEE International Conference on Fuzzy Systems (IEEE FUZZ-92),* 1153-1162, March 1992.

[16] Kosko, B., "Optimal Fuzzy Rules Cover Extrema," International *Journal of Intelligent Systems,* Vol. 10, No. 2, 249-255, February 1995.

[17] Kosko, B., "Combining Fuzzy Systems," *Proceedings of the IEEE* International *Conference on Fuzzy Systems (IEEE FUZZ-95),* Vol. IV, 1855-1863, March 1995.

[18] Kuo, B. *C., Automatic Control* Systems, 4th edition, Prentice Hall, Englewood Cliffs, NJ, 1982.

[19] Sheikholesam, S., and Desoer, C. A., "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] Specht, D. F. , "A General Regression Neural Network," *IEEE Transactions on Neural Networks,* Vol. 2, No. 6, 568-576, 1991.

[21] Sugeno, M., "An Introductory Survey of Fuzzy Control," *Information Sciences,* Vol. 36, 59-83, 1985.

Appendix I: SAM Theorem

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

$$F(x) = \frac{\sum_{j=1}^{m} a_j(x) V_j \; \mathbf{c_j}}{\sum_{j=1}^{m} a_j(x) V_j} = \sum_{j=1}^{m} p_j(x) \; \mathbf{c_j} \quad . \tag{41}$$

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

$$p_j(x) = \frac{a_j(x) V_j}{\sum_{k=1}^{m} a_k(x) V_k} \tag{42}$$

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

$$V_j = \int_{R^p} b_j(y_1, \ldots, y_p) dy_1 \cdots dy_p > \; 0 \tag{43}$$

$$\mathbf{c_j} = \frac{\int_{R^p} y \; b_j(y_1, \ldots, y_p) dy_1 \cdots dy_p}{\int_{R^p} b_j(y_1, \ldots, y_p) dy_1 \cdots dy_p} \tag{44}$$

Proof. There is no loss of generality to prove the theorem for the scalar-output case $p = 1$ when $F: R^n \to R^p$. 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 (43) and (44) as

$$V_j = \int_{-\infty}^{\infty} b_j(y) dy \tag{45}$$

$$c_j = \frac{\int_{-\infty}^{\infty} \mathrm{Y} \; b_j(y) dy}{\int_{-\infty}^{\infty} b_j(y) dy} \tag{46}$$

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

$$F(a) = Centroid(B) \tag{47}$$

$$= \frac{\int_{-\infty}^{\infty} y\, b(y)dy}{\int_{-\infty}^{\infty} b(y)dy} \tag{48}$$

$$= \frac{\int_{-\infty}^{\infty} y \sum_{j=1}^{m} b'_j(y)dy}{\int_{-\infty}^{\infty} \sum_{j=1}^{m} b'_j(y)dy} \tag{49}$$

$$= \frac{\int_{-\infty}^{\infty} y \sum_{j=1}^{m} a_j(x) b_j(y)dy}{\int_{-\infty}^{\infty} \sum_{j=1}^{m} a_j(x) b_j(y)dy} \tag{50}$$

$$= \frac{\sum_{j=1}^{m} a_j(x) \int_{-\infty}^{\infty} y\, b_j(y)dy}{\sum_{j=1}^{m} a_j(x) \int_{-\infty}^{\infty} b_j(y)dy} \tag{51}$$

$$= \frac{\sum_{j=1}^{m} a_j(x) V_j \frac{\int_{-\infty}^{\infty} y\, b_j(y)dy}{V_j}}{\sum_{j=1}^{m} a_j(x) V_j} \tag{52}$$

$$= \frac{\sum_{j=1}^{m} a_j(x) V_j\ c_j}{\sum_{j=1}^{m} a_j(x)\ V_j} \tag{53}$$

Q.E.D.

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

$$F(x) = Centroid(\sum_{j=1}^{m} w_j a_j(x)\ \ B_j) \tag{54}$$

$$= \frac{\sum_{j=1}^{m} w_j a_j(x) V_j\ c_j}{\sum_{j=1}^{m} w_j a_j(x)\ v_j} \tag{55}$$

The weight w_j has the same form as the area or volume weight V_j. We use $w_1 = \ldots = w_m > 0$ in the paper. So the weighted sum (55) reduces to (41). The centroidal structure (47)-(48) of the fuzzy system shows that *all* centroidal fuzzy systems compute a conditional expectation:

$$F(x) = \frac{\int_{-\infty}^{\infty} {}_{Y}\, b(x,y)dy}{\int_{-\infty}^{\infty} b(x,y)dy} \tag{56}$$

$$= \int_{-\infty}^{\infty} yp(y|x)dy \tag{57}$$

$$= E[Y|X = x] \tag{58}$$

since

$$p(y|x) = \frac{b(x,y)}{\int_{-cc}^{\infty} b(x,y)dy} \tag{59}$$

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

The SAM ratio (41) reduces to Sugeno's [21] "center of gravity" model

$$F(x) = \frac{\sum_{j=1}^{m} a_j(x)\ P_j}{\sum_{j=1}^{m} a_j(x)} \tag{60}$$

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 $V_1 = \cdots = V_m > 0$.

Appendix II: 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 product space X x 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 [2]-[3] tend to quickly move rules 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 first set of fuzzy rule patches in position and number and to initialize the gradient descents of supervised learning.

Learning changes system parameters with data. Unsupervised learning amounts to blind clustering in the system product space X x 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 $k >> m$.

A natural way to grow and tune rules is to identify a rule patch with the uncertainty ellipsoid [1]-[2] that forms around each quantizing vector q_j from the inverse of its positive definite covariance matrix K_j. Then sparse or noisy data grows 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:

$$\|z(t) - q_j(t)\| = \min(\|z(t) - q_1(t)\|, \ldots, \|z(t) - q_k(t)\|) \tag{61}$$

$$q_i(t+1) = \begin{cases} q_j(t) + \mu_t[z(t) - q_j(t)] & \text{if } i = j \\ q_i(t) & \text{if } i \neq j \end{cases} \tag{62}$$

$$K_i(t+1) = \begin{cases} K_j(t) + v_t[(z(t) - q_j(t))^T(z(t) - q_j(t)) - K_j(t)] & \text{if } i = j \\ K_i(t) & \text{if } i \neq j \end{cases} \tag{63}$$

for the Euclidean norm $\|z\|^2 = z_1^2 + \cdots + z_{n+p}^2$.

The first step (61) 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 [1]-[5] can then convert the ellipsoids into coordinate fuzzy sets. Other schemes can use the unfactored 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 $\mu_t \approx \frac{1}{t}$. The covariance coefficients obey a like constraint as in our choice of $v_t = 0.2[1 - \frac{t}{1.2N}]$ where N is the total number 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: $\varepsilon_t = d_t - F(x_t)$. 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 [5],[17] by changing the rule weights w_j in (64), 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

$$F(x) = \frac{\sum_{j=1}^{m} w_j \, a_j(x) V_j \, \mathbf{c_j}}{\sum_{j=1}^{m} w_j \, a_j(x) \, V_j} \tag{64}$$

in the same way as does the then-part volume V_j in the SAM Theorem. Both cancel from (53) if they have the same value-if $w_1 = \ldots = w_m > 0$ or if $V_1 = \ldots = V_m > 0$. So both have the same

learning law if we replace the nonzero weight w_j with the nonzero volume V_j or V_j':

$$w_j(t+1) = w_j(t) - \mu_t \frac{\partial E_t}{\partial w_j} \tag{65}$$

$$= w_j(t) - \mu_t \frac{\partial E_t}{\partial F} \frac{\partial F}{\partial w_j} \tag{66}$$

$$= w_j(t) + \mu_t \varepsilon_t \frac{p_j(x_t)}{w_j(t)}[c_j - F(x_t)] \tag{67}$$

for instantaneous squared error $E_t = \frac{1}{2}(d_t - F(x_t))^2$ with desired-minus-actual error $\varepsilon_t = d_t - F(x_t)$. We include the rule weights here for completeness. Our fuzzy systems were unweighted and thus used $w_1 = \ldots = w_m > 0$. 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):

$$V_j(t+1) = V_j(t) - \mu_t \frac{\partial E_t}{\partial V_j} \tag{68}$$

$$= V_j(t) + \mu_t \varepsilon_t \frac{p_j(x_t)}{V_j(t)}[c_j - F(x_t)] \tag{69}$$

The learning law (67) follows since $\frac{\partial E_t}{\partial w_j} = -\varepsilon$ and since

$$\frac{\partial F}{\partial w_j} = \frac{a_j(x)\, V_j\, c_j \sum_{i=1}^{m} w_i\ a_i(x)\ V_i - a_j(x) V_j \sum_{i=1}^{m} w_i\ a_i(x)\ V_i\, c_i}{(\sum_{i=1}^{m} w_i\, a_i(x)\, V_i)^2} \tag{70}$$

$$= \frac{w_j\ a_j(x)\ V_j}{w_j \sum_{i=1}^{m} w_i\ a_i(x)\ V_i} \left[\frac{c_j \sum_{i=1}^{m} w_i\, a_i(x) V_i}{\sum_{i=1}^{m} w_i\, a_i(x)\, V_i} - \frac{\sum_{i=1}^{m} w_i\, a_i(x)\, V_i\, c_i}{\sum_{i=1}^{m} w_i\, a_i(x)\, V_i}\right] \tag{71}$$

$$= \frac{p_j(x)}{w_j}[c_j - F(x)] \tag{72}$$

from the SAM Theorem.

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

$$c_j(t+1) = c_j(t) - \mu_t \frac{\partial E_t}{\partial F} \frac{\partial F}{\partial c_j} \tag{73}$$

$$= c_j(t) + \mu_t\, \varepsilon_t\, p_j(x_t) \quad . \tag{74}$$

So the terms w_j, V_j, and c_j do not change when $p_j \approx 0$ and thus when the jth 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 if-part set function a_j is a function of l parameters: $a_j = a_j(\text{mf}, \ldots, m_j^l)$. Then we can update each parameter with

$$m_j^k(t+1) = m_j^k(t) - \mu_t \frac{\partial E_t}{\partial F} \frac{\partial F}{\partial a_j} \frac{\partial a_j}{\partial m_j^k} \tag{75}$$

$$= m_j^k(t) + \mu_t \, \varepsilon_t \, \frac{p_j(x_t)}{a_j(x_t)} [c_j - F(x_t)] \frac{\partial a_j}{\partial m_j^k} \tag{76}$$

Exponential if-part set functions can reduce the learning complexity. They have the form $a_j = e^{f_j(m_j^1, \ldots, m_j^l)}$ and obey $\frac{\partial a_j}{\partial m_j^k} = a_j \frac{\partial f_j(m_j^1, \ldots, m_j^l)}{\partial m_j^k}$. Then the parameter update (76) simplifies to

$$m_j^k(t+1) = m_j^k(t) + \mu_t \, \varepsilon_t \, p_j(x_t)[c_j - F(x_t)] \frac{\partial f_j}{\partial m_j^k} \tag{77}$$

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

The Gaussian set function $a_j(x) = \exp\{-\frac{1}{2}\sum_{i=1}^{n}(\frac{x_i - m_j^i}{\sigma_j^i})^2\}$ has mean partial derivative $\frac{\partial f_j}{\partial m_j^k} = \frac{x_k - m_j^k}{(\sigma_j^k)^2}$ and variance partial derivative $\frac{\partial f_j}{\partial \sigma_j^k} = \frac{(x_k - m_j^k)^2}{(\sigma_j^k)^3}$. Such Gaussian set functions reduce the SAM model to Specht's [20] radial basis function network. We can use the smooth update law (77) to update non-differentiable triangles or trapezoids or other sets by viewing their centers and widths as the Gaussian means and variances.