Adaptive Fuzzy Systems for Backing up a Truck-and-Trailer

Seong-Gon Kong and Bart Kosko, *Member, IEEE*

Abstract— **This paper develops fuzzy control systems and neural-network control systems for backing up a simulated truck, and truck-and-trailer, to a loading dock in a parking lot. The supervised back-propagation learning algorithm trained the neural network systems. We tested the robustness of the neural systems by removing random subsets of training data in learning sequences. The neural systems performed well but required extensive computation for training. The fuzzy systems performed well until we removed over 50% of their fuzzy-associative-memory (FAM) rules. They also performed well when we replaced the key FAM equilibration rule with destructive, or "sabotage," rules. Unsupervised differential competitive learning (DCL) and product-space clustering adaptively generated FAM rules from training data. The original fuzzy control systems and neural control systems generated trajectory data. The DCL system rapidly recovered the underlying FAM rules. Product-space clustering converted the neural truck systems into structured sets of FAM rules that approximated the neural system's behavior.**

I. Fuzzy and Neural Control Systems

We construct fuzzy and neural control systems directly from control data, but from different types of control data. Fuzzy systems use a small number of structured *linguistic* input–output samples from an expert or from some other adaptive estimator. Neural systems use a large number of *numeric* input–output samples from the control process or from some other data base. Adaptive fuzzy systems also use numeric control data. Fig. 1. illustrates this difference. The neural system estimates function $f: X \rightarrow Y$ from several numerical *point* samples (x_i, y_i). The fuzzy system estimates f from a few fuzzy *set* samples or fuzzy associations (A_i, B_i).

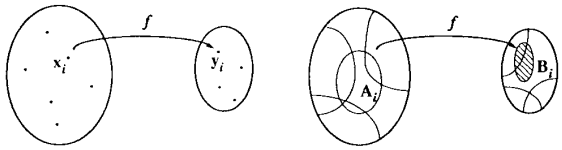
Fuzzy and neural systems offer a key advantage over traditional control approaches. They offer *model-free estimation* of the control system. The user need not specify how the controller's output mathematically depends on its input. Instead, the user provides a few common-sense associations of how the control variables behave. Or the user provides a statistically representative set of numerical training samples. Even if a math-model controller is available, fuzzy or neural controllers may prove more robust and easier to modify.

Manuscript received January 15, 1991; revised July 3, 1991. This work was supported by the Air Force Office of Scientific Research (AFOSR-88-0236) and by a grant for the Rockwell Science Center. Part of the material in this paper was presented at the 1990 International Joint Conference on Neural Networks, San Diego, CA, June 17–21, 1990.

S.-G. Kong was with the Department of Electrical Engineering, University of Southern California, Los Angeles, CA. He is now with the Department of Electrical Engineering, Soongshil University, Seoul, Korea.

B. Kosko is with the Department of Electrical Engineering, Signal and Image Processing Institute, University of Southern California, Los Angeles, CA 90089-2564.

IEEE Log Number 9104405.



(a) (b)

Fig. 1. Geometry of neural and fuzzy function estimation. The neural approach (a) uses several numerical point samples. The fuzzy approach (b) uses a few fuzzy set samples.

The type of system, fuzzy or neural, that performs better for a particular control problem depends on the type and availability of sample data. If experts provide structured knowledge of the control process or if sufficient numerical training samples are unavailable, the fuzzy approach may be preferable. We can construct a fuzzy control system with comparative ease when experts or fuzzy engineers provide accurate structured knowledge. A fuzzy control system seems a reasonable benchmark in such cases, even if we can develop a neural controller or math-model controller.

If we have representative numerical data but not structured expertise, the neural approach may be perferable. Or a statistical regression approach may be more appropriate. The data simply tell their own story—if there is a story to tell. Yet even here we can use a hybrid fuzzy-neural system, an adaptive fuzzy system. We can use the numerical data to generate *fuzzy associative memory* (FAM) rules. Each FAM rule defines a patch in the input–output state space, and the fuzzy system approximates the unknown function by covering its graph with FAM-rule patches [4]. The FAM rules can then form the skeleton of a fuzzy control *architecture*. In short, if structured knowledge is unavailable, estimate it. This may be more practical than it would appear because of the small number of control FAM rules needed to reliably control many real-world processes.

How can we compare fuzzy and neural controllers? Abstract comparison proves difficult because both approaches build a control black box in different ways. That they build black boxes distinguishes them from math-model controllers. It also suggests we can compare them, at least approximately, by their black-box control performance.

Each control system generates an output *control surface* as it ranges over the common input space of parameter values. Fig. 5 below shows three-dimensional control surfaces for the fuzzy and neural controllers. For control systems with

TABLE I
FUZZY SET VALUES OF THE FUZZY VARIABLES ϕ, x, AND θ

ϕ		x		θ	
RB	Right below	*LE*	Left	*NB*	Negative Big
RU	Right Upper	*LC*	Left Center	*NM*	Negative Medium
RV	Right Vertical	*CE*	Center	*NS*	Negative Small
VE	Vertical	*RC*	Right Center	*ZE*	Zero
LV	Left Vertical	*RI*	Right	*PS*	Positive Small
LU	Left Upper			*PM*	Positive Medium
LB	Left Below			*PB*	Positive Big

few input parameters with moderately quantized ranges, we can store both fuzzy and neural controllers—or rather their quantized control surfaces—as decision lookup tables. Then once we specify a system performance criterion, we can in principle quantitatively compare the controllers.

Comparing system trajectories proved more complicated. In the case at hand, we wanted to back up a truck and a truck-and-trailer to a loading dock. We can measure and compare the quality and quantity of the truck trajectory, perhaps with mean-squared error criteria. Intuitively, we preferred smooth, short trajectories to jagged, long trajectories. Reaching the loading-dock goal was also important. In practive it is the most important performance requirement. We must balance the trajectory type with the trajectory destination, and this reduces to the pragmatic issue of balancing means and ends.

Below we develop a simple fuzzy control system and a simple neural control system for backing up a truck and a truck-and-trailer in an open parking lot. The recent neural network truck backer-upper simulation of Nguyen and Widrow [7] motivated our choice of control problem.

The fuzzy control system compared favorably with the neural controller in terms of black-box development effort, black-box computational load, smoothness of truck trajectories, and robustness.

We studied robustness of the fuzzy control systems in two ways. We deliberately added confusing FAM rules—"sabotage" rules—to the system, and we randomly removed different subsets of FAM rules. We studied robustness of the neural controller by randomly removing different portions of the training data in learning sequences. We also converted the neural control systems to structured FAM-bank systems.

II. TRUCK BACKER-UPPER CONTROL SYSTEMS

A. Backing up a Truck

Fig. 2 shows the geometry of the simulated truck and loading dock. The truck corresponds to the cab part of the truck-trailer in the Nguyen–Widrow neural truck backer-upper. The three state variables ϕ, x, and y determine the truck position with ϕ specifying the angle of the truck with the horizontal. The coordinate pair (x, y) specifies the position of the rear center of the truck.

We wanted the truck to arrive at the loading dock at a right angle ($\phi_f = 90°$) and to align the position (x, y) of the truck with the desired loading dock (x_f, y_f). We considered only backing up. The truck moved backward a fixed distance at every stage until the truck hits the border of the loading zone. The loading zone corresponded to the plane $[0, 100] \times [0, 100]$, and (x_f, y_f) equaled $(50, 100)$.

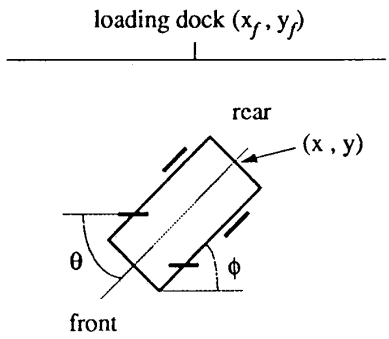


Fig. 2. Diagram of simulated truck and loading dock.

At every stage the fuzzy and neural controllers should produce the steering angle, θ, that backs up the truck to the loading dock from any initial position and from any angle in the loading zone.

B. Fuzzy Truck Backer-Upper System

We first specified the control as a function of the state. The state variables were the truck angle, ϕ, and the truck x-position coordinate, x. The control variable was the steering-angle signal, θ. We assumed enough clearance between the truck and the loading dock so we could ignore the truck y-position coordinate, y. The coordinate x ranges from 0 to 100, ϕ ranges from -90 to 270, and θ ranges from -30 to 30. Positive values of θ represented clockwise rotations of the steering wheel, and negative values represented counterclockwise rotations. We discretized all values to reduce computation. The resolution of ϕ and θ was 1° each. The resolution of x was 0.1.

Next we specified the fuzzy-set values of the state and control fuzzy variables. The fuzzy sets represented numerical values for linguistic terms, the sort of linguistic terms an expert might use to describe the control system. We chose the fuzzy-set values of the fuzzy variables as in Table I.

Fuzzy subsets contain elements with degrees of membership. A fuzzy membership function $m_F : Z \rightarrow [0, 1]$ assigns a real number between 0 and 1 to every element z in the universe of discourse Z. This number $m_F(z)$ indicates the degree to which the object or data z belongs to the fuzzy set

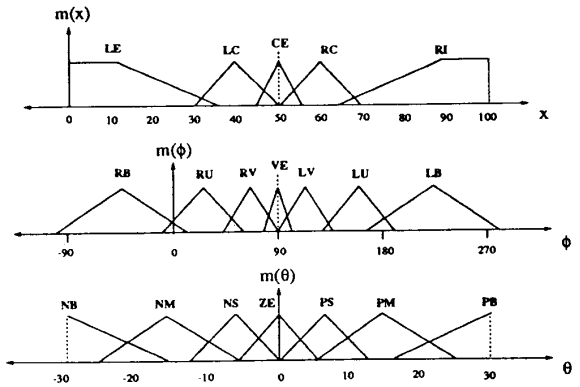


Fig. 3. Fuzzy membership functions for each linguistic fuzzy-set value. To allow finer control, the fuzzy sets that correspond to near the loading dock are narrower than the fuzzy sets that correspond to far from the loading dock.

F. Equivalently, $m_F(z)$ defines the *fit* (fuzzy unit) value [3] of element z in F.

Fuzzy membership functions can have different shapes, depending on the designer's preference or experience. In practice fuzzy engineers have found that triangular and trapezoidal shapes simplify computation and help capture the modeler's sense of fuzzy numbers. Fig. 3 shows membership-function graphs of the fuzzy subsets above. In the third graph, for example, $\theta = 20°$ is positive medium to degree 0.5, but positive big only to degree 0.3.

In Fig. 3 the fuzzy sets CE, VE, and ZE are narrower than the other fuzzy sets. These narrow fuzzy sets permit fine control near the loading dock. We used wider fuzzy sets to describe the endpoints of the range of the fuzzy variables ϕ, x, and θ. The wider fuzzy sets permitted rough control far from the loading dock.

Next we specified the fuzzy "rule base," or bank of FAM rules. Fuzzy associations or "rules" (A, B) associate output fuzzy sets B of control values with input fuzzy sets A of input-variable values. We can write fuzzy associations as antecedent–consequent pairs or IF–THEN statements.

In the truck backer-upper case, the FAM bank contained the 35 FAM rules in Fig. 4. For example, FAM rule 1 $(LE, RB; PS)$ corresponds to the following association:

$$\text{IF } x = LE \quad \text{AND} \quad \phi = RB, \quad \text{THEN } \theta = PS.$$

FAM rule 18 indicates that if the truck is near the equilibrium position, then the controller should not produce a positive or negative steering-angle signal. The FAM rules in the FAM-bank matrix reflect the symmetry of the controlled system.

For the initial condition $x = 50$ and $\phi = 270$, the fuzzy truck did not perform well. The symmetry of the FAM rules and the fuzzy sets canceled the fuzzy controller output in a rare saddle point. For this initial condition, the neural controller (and truck-and-trailer below) also performed poorly. Any perturbation breaks the symmetry. For example, the rule (if $x = 50$ and $\phi = 270$, then $\theta = 5$) corrected the problem.

The three-dimensional control surfaces in Fig. 5 show steering-angle signal outputs θ that correspond to all

ϕ \ x	LE	LC	CE	RC	RI
RB	[1] PS	[2] PM	[3] PM	[4] PB	[5] PB
RU	[6] NS	[7] PS	PM	PB	PB
RV	NM	NS	PS	PM	PB
VE	NM	NM	[18] ZE	PM	PM
LV	NB	NM	NS	PS	PM
LU	NB	NB	NM	NS	PS
LB	NB	NB	NM	NM	[35] NS

Fig. 4. FAM-bank matrix for the fuzzy truck backer-upper controller.

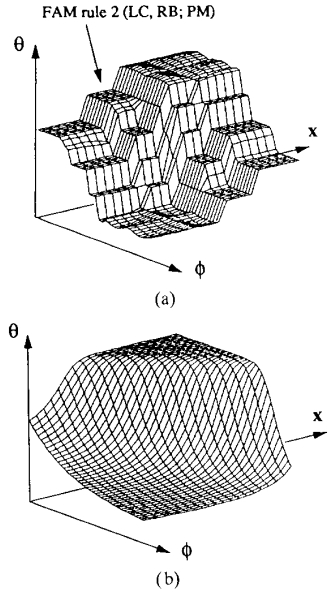


Fig. 5. (a) Control surface of the fuzzy controller. Fuzzy-set values determined the input and output combination corresponding to FAM rule 2 (IF $x = LC$ AND $\phi = RB$, THEN $\theta = PM$). (b) Corresponding control surface of the neural controller for constant value $y = 20$.

combinations of values of the two input state variables ϕ and x. The control surface defines the fuzzy controller. In this simulation the correlation-minimum FAM inference procedure, discussed in [4], determined the fuzzy control surface. If the control surface changes with sampled variable values, the system behaves as an *adaptive* fuzzy controller. Below we demonstrate unsupervised adaptive control of the truck and the truck-and-trailer systems.

Finally, we determined the output action given the input conditions. We used the correlation-minimum inference method illustrated in Fig. 6. Each FAM rule produced the output fuzzy set clipped at the degree of membership determined by the input conditions and the FAM rule. Alternatively, correlation-product inference [4] would combine FAM rules multiplicatively. Each FAM rule emitted a fit-weighted output

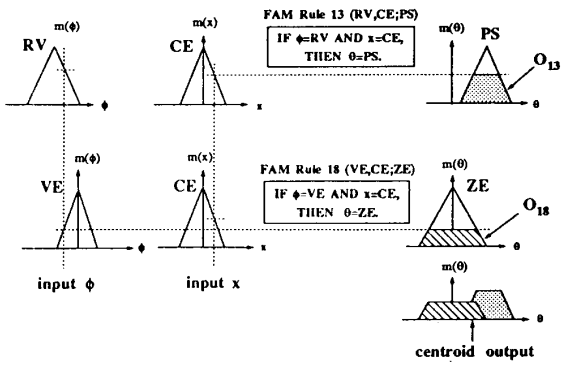


Fig. 6. Correlation-minimum inference with centroid defuzzification method. Then FAM-rule antecedents combined with AND use the *minimum* fit value to activate consequents. Those combined with OR would use the *maximum* fit value.

fuzzy set, O_i, at each iteration. The total output, O, added these weighted outputs:

$$O = \sum_i O_i \tag{1}$$

$$= \sum_i \min(f_i, S_i), \tag{2}$$

where f_i denotes the antecedent fit value, and S_i represents the consequent fuzzy set of steering-angle values in the ith FAM rule. Earlier fuzzy systems combined the output sets, O_i, with pairwise maxima. But this tends to produce a uniform output set O as the number of FAM rules increases. Adding the output sets O_i invokes the fuzzy version of the central limit theorem. This tends to produce a symmetric, unimodal output fuzzy set O of steering-angle values.

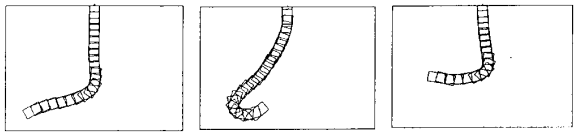
Fuzzy systems map fuzzy sets to fuzzy sets. The fuzzy control system's output defines the fuzzy set O of steering-angle values at each iteration. We must "defuzzify" the fuzzy set O to produce a numerical (point-estimate) steering-angle output value θ.

As discussed in [4], the simplest defuzzification scheme selects the value corresponding to the *maximum fit* value in the fuzzy set. This mode-selection approach ignores most of the information in the output fuzzy set and requires an additional decision algorithm when multiple modes occur.

Centroid defuzzification provides a more effective procedure. This method uses the *fuzzy centroid,* $\overline{\theta}$, as output:

$$\overline{\theta} = \frac{\sum_{j=1}^{p} \theta_j m_O(\theta_j)}{\sum_{j=1}^{p} m_O(\theta_j)}, \tag{3}$$

where O defines a fuzzy subset of the steering-angle universe of discourse $\Theta = \{\theta_1, \cdots, \theta_p\}$. The central-limit-theorem effect produced by adding output fuzzy set O_i benefits both max-mode and centroid defuzzification. Fig. 6 shows the correlation-minimum inference and centroid defuzzification applied to FAM rules 13 and 18. We used centroid defuzzification in all simulations.



(a) (b) (c)

Fig. 7. Sample truck trajectories of the fuzzy controller for initial positions (x, y, ϕ): (a) (20, 20, 30), (b) (30, 10, 220), and (c) (30, 40, −10).

With 35 FAM rules, the fuzzy truck controller produced successful truck backing-up trajectories starting from any initial position. Fig. 7 shows typical examples of the fuzzy-controlled truck trajectories from different initial positions. The fuzzy control system did not use ("fire") all FAM rules at each iteration. Equivalently most output consequent sets are empty. In most cases the system used only one or two FAM rules at each iteration. The system used at most four FAM rules at once.

C. Neural Truck Backer-Upper System

The neural truck backer-upper of Nguyen and Widrow [7] consisted of multilayer feedforward neural networks trained with the back-propagation gradient-descent (stochastic-approximation) algorithm. The *neural control system* consisted of two neural networks: the controller network and the truck emulator network. The *controller network* produced an appropriate steering-angle signal output given any parking-lot coordinates (x, y) and the angle ϕ. The *emulator network* computed the next position of the truck. The emulator network took as input the previous truck position and the current steering-angle output computed by the controller network.

We did not train the emulator network since we could not obtain "universal" synaptic connection weights for the truck emulator network. The back-propagation learning algorithm did not converge for some sets of training samples. The number of training samples for the emulator network might exceed 3000. For example, the combinations of training samples of a given angle ϕ, x position, y position, and steering angle signal θ might correspond to 3150 ($18 \times 5 \times 5 \times 7$) samples, depending on the division of the input–output product space. Moreover, the training samples were numerically similar since the neuronal signals assumed scaled values in $[0, 1]$ or $[-1, 1]$. For example, we treated close values, such as 0.40 and 0.41, as distinct sample values.

Simple kinematic equations replaced the truck emulator network. If the truck moved backward from (x, y) to (x', y') at an iteration, then

$$x' = x + r\cos(\phi') \tag{4}$$

$$y' = y + r\sin(\phi') \tag{5}$$

$$\phi' = \phi + \theta. \tag{6}$$

Here r denotes the fixed driving distance of the truck for all backing movements. We used (4)–(6) instead of the emulator network. This did not affect the posttraining performance of the neural truck backer-upper since the truck emulator network back-propagated only errors.

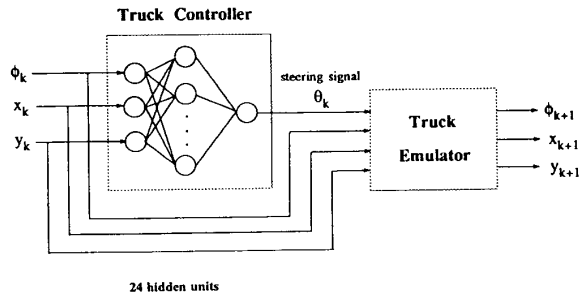


Fig. 8. Topology of our neural control system.

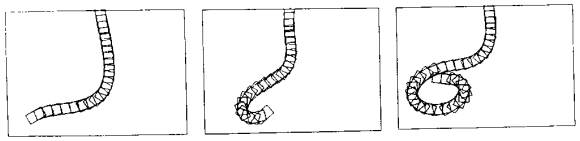


Fig. 9. Sample truck trajectories of the neural controller for initial positions (x, y, ϕ): (a) (20, 20, 30), (b) (30, 10, 220), and (c) (30, 40, -10).

We trained only the controller network with back-propatation. The controller network used 24 "hidden" neurons with logistic sigmoid functions. In the training of the truck controller, we estimated the ideal steering-angle signal at each stage before we trained the controller network. In the simulation, we used the arc-shaped truck trajectory produced by the fuzzy controller as the ideal trajectory. The fuzzy controller generated each training sample (x, y, ϕ, θ) at each iteration of the backing-up process. We used 35 training sample vectors needed more than 100 000 iterations to train the controller network.

Fig. 5(b) shows the resulting neural control surface for $y = 20$. The neural control surface shows less structure than the corresponding fuzzy control surface. This reflects the unstructured nature of black-box supervised learning. Fig. 8 shows the network connection topology for our neural truck backer-upper control system.

Fig. 9 shows typical examples of the neural-controlled truck trajectories from several initial positions. Even though we trained the neural network to follow the smooth arc-shaped path, some learned truck trajectories were nonoptimal.

D. Comparison of Fuzzy and Neural Systems

As shown in Figs. 7 and 9, the fuzzy controller always smoothly backed up the truck but the neural controller did not. The neural-controlled truck sometimes followed an irregular path.

Training the neural control system was time-consuming. The back-propagation algorithm required thousands of back-ups to train the controller network. In some cases, the learning algorithm did not converge.

We "trained" the fuzzy controller by encoding our own common-sense FAM rules. Once we develop the FAM-rule bank, we can compute control outputs from the resulting FAM-bank matrix or control surface. The fuzzy controller did not need a truck emulator and did not require a math model of how outputs depended on inputs.

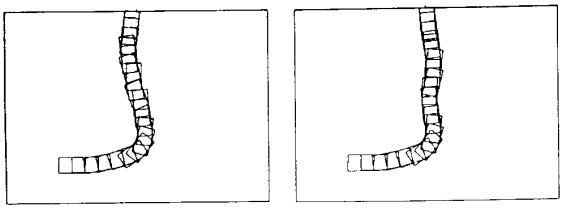


Fig. 10 The fuzzy truck trajectory after we replaced the key steady-state FAM rule 18 by the two worst rules: (a) IF $x = CE$ AND $\phi = VE$, THEN $\theta = PB$, and (b) IF $x = CE$ AND $\phi = VE$, THEN $\theta = NB$.

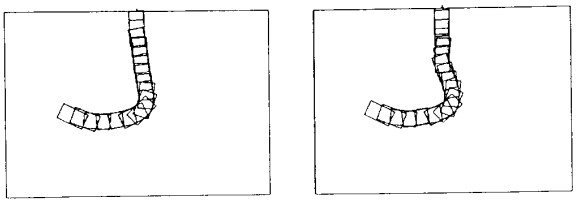


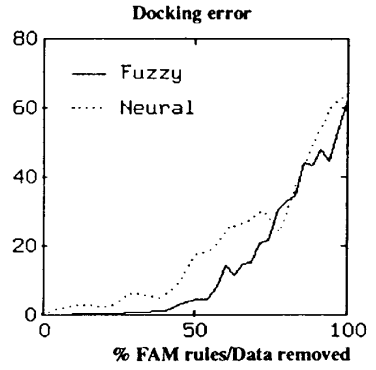
Fig. 11. Fuzzy truck trajectory when (a) no FAM rules are removed and (b) FAM rules 7, 13, 18, and 23 are removed.

The fuzzy controller was computationally lighter than the neural controller. Most computation operations in the neural controller involved the multiplication, addition, or logarithm of two real numbers. In the fuzzy controller, most computational operations involved comparing and adding two real numbers.

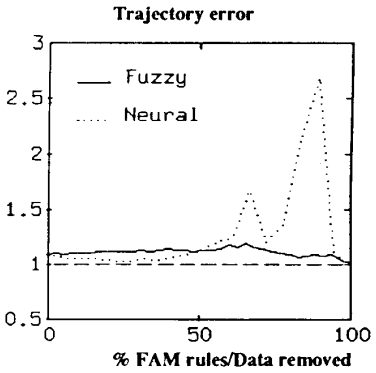
E. Sensitivity Analysis

We studied the sensitivity of the fuzzy controller in two ways. We replaced the FAM rules with destructive, or "sabotage," FAM rules, and we randomly removed FAM rules. We deliberately chose sabotage FAM rules to confound the system. Fig. 10 shows the trajectory when two sabotage FAM rules replaced the important steady-state FAM rule—FAM rule 18: the fuzzy controller should produce zero output when the truck is nearly in the correct parking position. Fig. 11 shows the truck trajectory after we removed four randomly chosen FAM rules (7, 13, 18, and 23). These perturbations did not significantly affect the fuzzy controller's performance.

We studied robustness of each controller by examining failure rates. For the fuzzy controller we removed fixed percentages of randomly selected FAM rules from the system. For the neural controller we removed training data. Fig. 12 shows performance errors averaged over ten typical back-ups with missing FAM rules for the fuzzy controller and missing training data for the neural controller. The missing FAM rules and training data ranged from 0% to 100% of the total. In Fig. 12(a), the docking error equaled the Euclidean distance from the actual final position (ϕ, x, y) to the desired final



(a)



(b)

Fig. 12. Comparison of the robustness of the controllers: (a) docking error and (b) trajectory error.

position (ϕ_f, x_f, y_f):

$$\text{docking error} = \sqrt{(\phi_f - \phi)^2 + (x_f - x)^2 + (y_f - y)^2}. \tag{7}$$

In Fig. 12(b), the trajectory error equaled the ratio of the actual trajectory length of the truck divided by the straight line distance to the loading dock:

$$\text{trajectory error} = \frac{\text{length of truck trajectory}}{\text{distance(initial position, desired final position)}}. \tag{8}$$

III. Adaptive Fuzzy Truck Backer-Upper Controller Systems

A. Stochastic Competitive Learning Algorithms

Product-space clustering [4] is a form of stochastic adaptive vector quantization. Adaptive vector quantization (AVQ) systems adaptively quantize pattern clusters in R^n. Stochastic competitive learning systems are neural AVQ systems. Neurons compete for the activation induced by randomly sampled patterns. The corresponding synaptic fan-in vectors adaptively

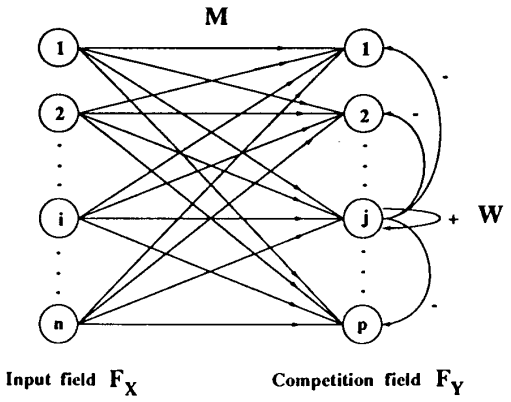
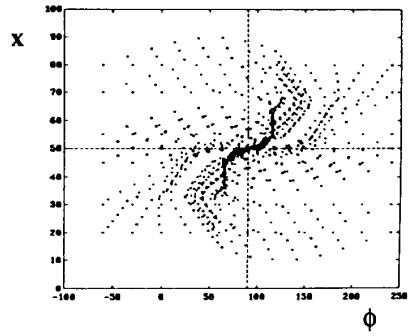
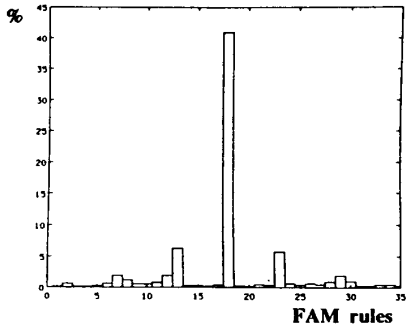


Fig. 13. Topology of the laterally inhibitive DCL network.



(a)



(b)

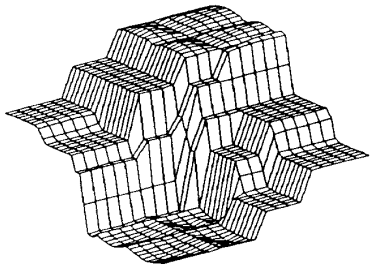
Fig. 14. (a) Input data distribution. (b) Synaptic-vector histogram. Differential competitive learning allocated synaptic quantization vectors to FAM cells. The steady-state FAM cell $(CE, VE; ZE)$ contained the most synaptic vectors.

quantize the pattern space R^n. The p synaptic vectors m_j define the p columns of the synaptic connection matrix M. M interconnects the n input or linear neurons in the input neuronal field, F_X, to the p competing nonlinear neurons in the output field, F_Y.

Learning algorithms estimate the unknown probability density function $p(x)$, which describes the distribution of patterns in R^n. More synaptic vectors arrive at more probable regions.

ϕ \ x	LE	LC	CE	RC	RI
RB	PS	PM	PM	PB	PB
RU	ZE	PM	PM	PB	PB
RV	NS	ZE	PS	PM	PB
VE	NM	NM	ZE	PM	PM
LV	NB	NB	NS	PS	PS
LU	NB	NB	NB	NS	PS
LB	NB	NB	NM	NM	NS

(a)

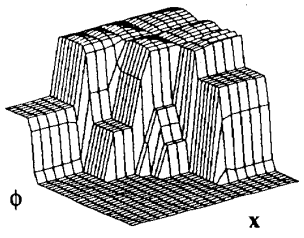


(b)

Fig. 15. (a) DCL-estimated FAM bank. (b) Corresponding control surface.

ϕ \ x	LE	LC	CE	RC	RI
RB	PS	PB	PB	PB	PB
RU	NM	ZE	PM	PB	PB
RV	NM	NM	NS	PS	PB
VE	NM	NM	NM	ZE	PB
LV	NM	NM	NM	NS	PB
LU	NM	NM	NM	NM	PM
LB	NM	NM	NM	NM	NM

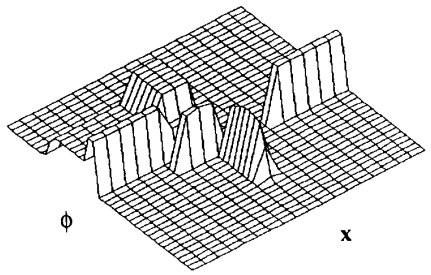
(a)



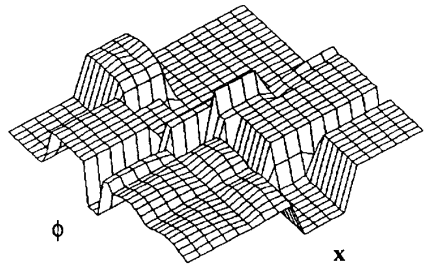
(b)

Fig. 16. (a) FAM bank generated by the neural control surface. (b) Control surface of the neural BP-AFAM system in (a).

Where sample vectors x are dense or sparse, synaptic vectors $\boldsymbol{m}_j$ should be dense or sparse. The local count of synaptic vectors then gives a nonparametric estimate of the volume



(a)



(b)

Fig. 17. (a) Absolute difference of the FAM surface in Fig. 5(a) and the DCL-estimated FAM surface in Fig. 14(b). (b) Absolute difference of the FAM surface in Fig. 5(a) and the neural-estimated FAM surface in Fig. 15(b).

probability $P(V)$ for volume $V \subset R^n$:

$$P(V) = \int_V p(\boldsymbol{x})\, d\boldsymbol{x} \tag{9}$$

$$\approx \frac{\text{number of } \boldsymbol{m}_j \in V}{p}. \tag{10}$$

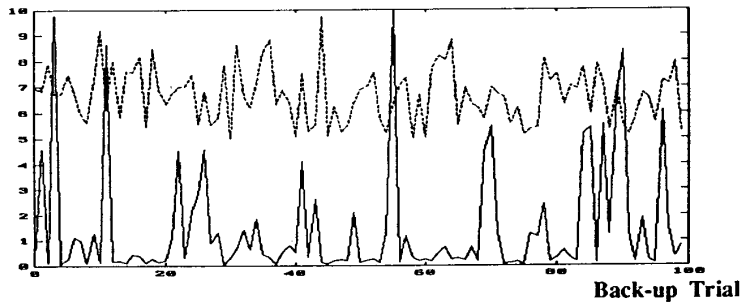
In the extreme case where $V = R^n$, this approximation gives $P(V) = p/p = 1$. For improbable subsets V, $P(V) = 0/p = 0$.

The metaphor of competing neurons reduces to nearest-neighbor classification. The AVQ system compares the current vector random sample $\boldsymbol{x}(t)$ in Euclidean distance to the p columns of the synaptic connection matrix M, with the p synaptic vectors $\boldsymbol{m}_1(t), \cdots, \boldsymbol{m}_p(t)$. If the jth synaptic vector $\boldsymbol{m}_j(t)$ is closest to $\boldsymbol{x}(t)$, then the jth output neuron "wins" the competition for activation at time t. In practice we sometimes define the nearest N synaptic vectors as winners. Some scaled form of $\boldsymbol{x}(t) - \boldsymbol{m}_j(t)$ updates the nearest or "winning" synaptic vectors. "Losers" remain unchanged: $\boldsymbol{m}_i(t+1) = \boldsymbol{m}_i(t)$. Competitive synaptic vectors converge to pattern-class centroids exponentially fast [5].

The following three-step process describes the competitive AVQ algorithm, where the third step depends on which learning algorithm updates the winning synaptic vectors.

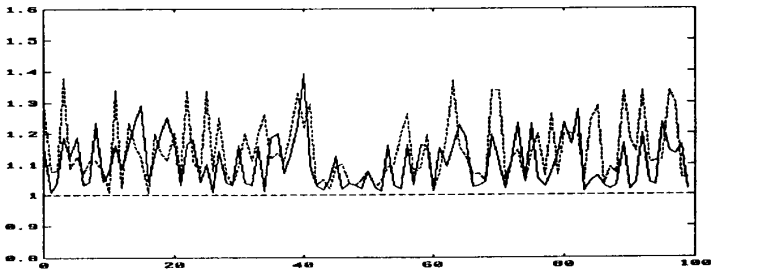
Competitive AVQ Algorithm

1) Initialize synaptic vectors: $\boldsymbol{m}_i(0) = \boldsymbol{x}(i)$, $i = 1, \cdots, p$. Sample-dependent initialization avoids many pathologies that can distort nearest-neighbor learning.



- DCL-AFAM (solid) : mean = 1.4449, s.d. = 2.2594
- BP-AFAM (dashed) : mean = 6.6863, s.d. = 1.0665

(a)



- DCL-AFAM (solid) : mean = 1.1075, s.d. = 0.0839
- BP-AFAM (dashed) : mean = 1.1453, s.d. = 0.1016

(b)

Fig. 18. Docking errors of the DCL-AFAM and BP-AFAM control systems: (a) docking error; (b) trajectory error.

2) For random sample $\boldsymbol{x}(t)$, find the closest or "winning" synaptic vector $\boldsymbol{m}_j(t)$:

$$\|\boldsymbol{m}_j(t) - \boldsymbol{x}(t)\| = \min_i \|\boldsymbol{m}_i(t) - \boldsymbol{x}(t)\|, \tag{11}$$

where $\|\boldsymbol{x}\|^2 = x_1^2 + \cdots + x_n^2$ defines the squared Euclidean vector norm of $\boldsymbol{x}$. We can define the N synaptic vectors closest to $\boldsymbol{x}$ as "winners".

3) Update the winning synaptic vector(s) $\boldsymbol{m}_j(t)$ with an appropriate learning algorithm.

B. Differential Competitive Learning

Differential competitive "synapses" learn only if the competing "neuron" changes its competitive status [6]:

$$\dot{m}_{ij} = \dot{S}_j(y_j)[S_i(x_i) - m_{ij}], \tag{12}$$

or, in vector notation,

$$\dot{\boldsymbol{m}}_j = \dot{S}_j(y_j)[\boldsymbol{S}(x) - \boldsymbol{m}_j], \tag{13}$$

where $\boldsymbol{S}(\boldsymbol{x}) = (S_1(x_1), \cdots, S_n(x_n))$ and $\boldsymbol{m}_j = (m_{1j}, \cdots, m_{nj})$. The quantity m_{ij} denotes the synaptic weight between the ith neuron in input field F_X and the jth neuron in competitive field F_Y. Nonnegative signal functions S_i and S_j transduce the real-valued activations x_i and y_j into bounded monotone nondecreasing signals $S_i(x_i)$ and $S_j(y_j)$. The symbols $\dot{m}_{ij}$ and $\dot{S}_j(y_j)$ denote the time derivatives of m_{ij} and $S_j(y_j)$, synaptic and signal velocities. $S_j(y_j)$ measures the competitive status of the jth competing neuron in F_Y. Usually S_j approximates a binary threshold function. For example, S_j may equal a steep binary logistic sigmoid,

$$S_j(y_j) = \frac{1}{1 + e^{-cy_j}}, \tag{14}$$

for some constant $c > 0$. The jth neuron wins the laterally inhibitive competition if $S_j = 1$ and loses if $S_j = 0$.

For discrete implementation, we use the DCL algorithm as a stochastic difference equation [2]:

$$\boldsymbol{m}_j(t+1) = \boldsymbol{m}_j(t) + c_t \Delta S_j(y_j(t))[S(\boldsymbol{x}(t)) - \boldsymbol{m}_j(t)] \quad \text{if the } j\text{th neuron wins} \tag{15}$$

$$\boldsymbol{m}_i(t+1) = \boldsymbol{m}_i(t) \quad \text{if the } i\text{th neuron loses.} \tag{16}$$

$\Delta S_j(y_j(t))$ denotes the time change of the jth neuron's competition signal $S_j(y_j)$ in the competitive field F_Y:

$$\Delta S_j(y_j(t)) = \operatorname{sgn}[S_j(y_j(t+1)) - S_j(y_j(t))]. \tag{17}$$

We define the signum operator $\operatorname{sgn}(x)$ as

$$\operatorname{sgn}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0. \end{cases} \tag{18}$$

The symbol $\{c_t\}$ denotes a slowly decreasing sequence of learning coefficients, such as $c_t = 0.1\,(1 - t/N)$ for

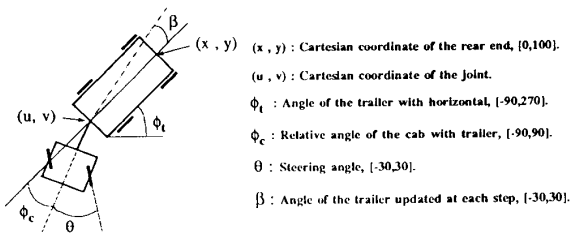


Fig. 19. Diagram of the simulated truck-and-trailer system.

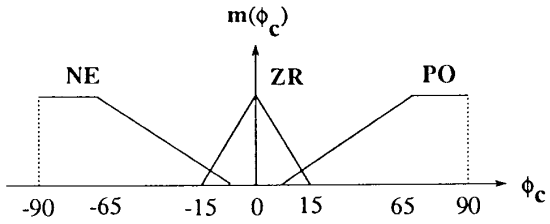


Fig. 20. Membership graphs of the three fuzzy-set values of fuzzy variable ϕ_c.

N training samples. Stochastic approximation [1] requires a decreasing gain sequence $\{c_t\}$ to suppress random disturbances and to guarantee convergence to local minima of mean-squared performance measures. The learning coefficients should decrease slowly,

$$\sum_{t=1}^{\infty} c_t = \infty, \tag{19}$$

but not too slowly,

$$\sum_{t=1}^{\infty} c_t^2 < \infty. \tag{20}$$

Harmonic-series coefficients, $c_t = 1/t$, satisfy these constraints.

We approximate the competitive signal difference ΔS_j as the activation difference Δy_j:

$$\Delta S_j(y_j(t)) = \operatorname{sgn}[y_j(t+1) - y_j(t)] \tag{21}$$

$$= \Delta y_j(t). \tag{22}$$

Input neurons in feedforward networks usually behave linearly: $S_i(x_i) = x_i$, or $S(\boldsymbol{x}(t)) = \boldsymbol{x}(t)$. Then we update the winning synaptic vector $\boldsymbol{m}_j(t)$ with

$$\boldsymbol{m}_j(t+1) = \boldsymbol{m}_j(t) + c_t \Delta y_j(t)[\boldsymbol{x}(t) - \boldsymbol{m}_j(t)]. \tag{23}$$

We update the F_Y neuronal activations y_i with the additive model

$$y_j(t+1) = y_j(t) + \sum_{i}^{n} S_i(x_i(t)) m_{ij}(t) + \sum_{k}^{p} S_k(y_k(t)) w_{kj}. \tag{24}$$

For linear signal functions S_i, the first sum in (24) reduces to an inner product of sample and synaptic vectors:

$$\sum_{i}^{n} x_i(t) m_{ij}(t) = \boldsymbol{x}^T(t) \boldsymbol{m}_j(t). \tag{25}$$

Then positive learning tends to occur ($\Delta m_{ij} > 0$) when $\boldsymbol{x}$ is close to the jth synaptic vector, $\boldsymbol{m}_j$.

The $p \times p$ matrix W contains the F_Y within-field synaptic connection strengths. Diagonal elements w_{ii} are positive, off-diagonal elements negative. Winning neurons excite themselves and inhibit all other neurons. Fig. 13 shows the connection topology of the laterally inhibitive DCL network.

C. Product-Space Clustering to Generate FAM Rules

Adaptive FAM (AFAM) systems generate FAM rules directly from training data. A one-dimensional FAM system, $S: I^n \rightarrow I^p$, defines a FAM rule, a single association of the form (A_i, B_i). In this case the input–output product space equals $I^n \times I^p$. As discussed in [4], a FAM rule (A_i, B_i) defines a patch or cluster or ball of points in the product-space cube $I^n \times I^p$ centered at the point (A_i, B_i). Adaptive clustering algorithms can estimate the unknown FAM rule (A_i, B_i) from training samples in R^2. We used differential competitive learning (DCL) to recover the bank of FAM rules that generated the truck training data.

We generated 2230 truck samples from seven different initial positions and varying angles. We chose the initial positions (20, 20), (30, 20), (45, 20), (50, 20), (55, 20), (70, 20), and (80, 20). We changed the angle from $-60°$ to $240°$ at each initial position. At each step, the fuzzy controller produced output steering angle θ. The training vectors (x, ϕ, θ) defined points in a three-dimensional product space. x had five fuzzy set values: LE, LC, CE, RC, and RI. ϕ had seven fuzzy set values: RB, RU, RV, VE, LV, LU, and LB. θ had seven fuzzy set values: NB, NM, NS, ZE, PS, PM, and PB. So there were 245 ($5 \times 7 \times 7$) possible FAM cells.

We defined FAM cells by partitioning the effective product space. FAM cells near the center were smaller than outer FAM cells because we chose narrow membership functions near the steady-state FAM cell. Uniform partitions of the product space produced poor estimates of the original FAM rules. As in Fig. 3, this reflected the need to judiciously define the fuzzy-set values of the system fuzzy variables.

We divided the space $0 \leq x \leq 100$ into five nonuniform intervals: $[0, 32.5]$, $[32.5, 47.5]$, $[47.5, 52.5]$, $[52.5, 67.5]$, and $[67.5, 100]$. Each interval represented the five fuzzy-set values LE, LC, CE, RC, and RI. This choice corresponded to the nonoverlapping intervals of the fuzzy membership function graphs $m(x)$ in Fig. 3. Similarly, we divided the space $-90 \leq \phi \leq 270$ into seven nonuniform intervals: $[-90, 0]$, $[0, 66.5]$, $[66.5, 86]$, $[86, 94]$, $[94, 113.5]$, $[133.5, 182.5]$, and $[182.5, 270]$, which corresponded, respectively, to RB, RU, RV, VE, LV, LU, and LB. We divided the space $-30 \leq \theta \leq 30$ into seven nonuniform intervals: $[-30, -20]$, $[-20, -7.5]$, $[-7.5, -2.5]$, $[-2.5, 2.5]$, $[2.5, 7.5]$, $[7.5, 20]$, and $[20, 30]$, which corresponded to NB, NM, NS, ZE, PS, PM, and PB.

DCL classified each input–output data vector into one of the FAM cells. We added a FAM rule to the FAM bank if the DCL-trained synaptic vector fell in the FAM cell. In the

We combined the steering-angle domain θ with the following for ϕ. ϕ ranged from $-90°$ to $90°$. We chose the fuzzy set

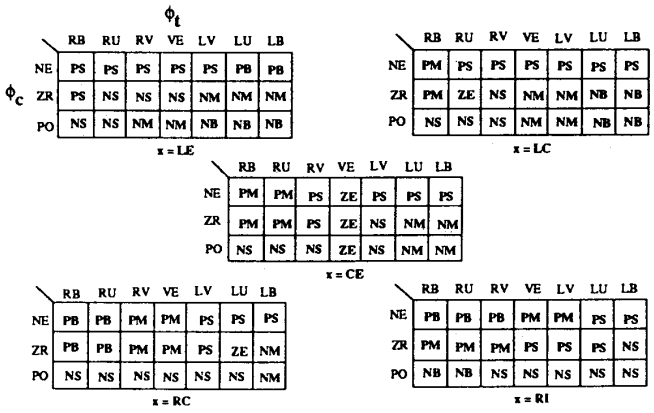
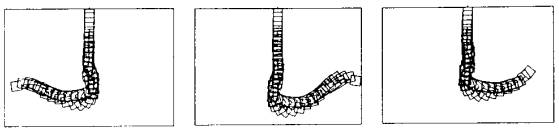
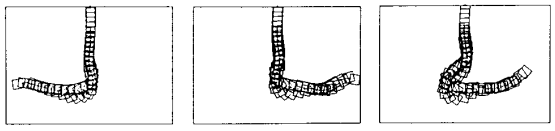


Fig. 21. FAM bank of the fuzzy truck-and-trailer control system.



(a) (b) (c)

Fig. 22. Sample truck-and-trailer trajectories from the fuzzy controller for initial positions (x, y, ϕ_t, ϕ_c): (a) (25, 30, −20, 30), (b) (80, 30, 210, −40), and (c) (70, 30, 200, 30).



(a) (b) (c)

Fig. 23. Sample truck-and-trailer trajectories of the BP-trained controller for initial positions (x, y, ϕ_t, ϕ_c): (a) (25, 30, −20, 30), (b) (80, 30, 210, −40), and (c) (70, 30, 200, 30).

case of ties we chose the FAM cell with the most densely clustered data.

For the BP-AFAM generated from the neural control surface in Fig. 16, we divided the rectangle $[0, 100] \times [-90, 270]$ into 35 nonuniform squares with the same divisions defined above. Then we added and averaged the control surface values in the square. We added a FAM rule to the FAM bank if the averaged value correponded to one of the seven FAM cells.

Fig. 14(a) shows the input sample distribution of (x, ϕ). We did not include the variable θ in the figure. Training data clustered near the steady-state position ($x = 50$ and $\phi = 90°$). Fig. 14(b) displays the synaptic-vector histogram after DCL classified 2230 training vectors for 35 FAM rules. Since successful FAM system generated the training samples, most training samples, and thus most synaptic vectors, clustered in the steady-state FAM cell.

DCL product-space clustering estimated 35 new FAM rules. Fig. 15 shows the DCL-estimated FAM bank and the corresponding control surface. The DCL-estimated control surface visually resembles the underlying unknown control surface in Fig. 5. The two systems produce nearly equivalent truck-backing behavior. This suggests that adaptive product-space clustering can estimate the FAM rules underlying expert behavior in many cases, even when the expert or fuzzy engineer cannot articulate the FAM rules.

We also used the neural control surface in Fig. 5(b) to estimate FAM rules. We divided the input−output product space into FAM cells as in the fuzzy control case. If the neural control surface intersected the FAM cell, we entered the corresponding FAM rule in a FAM bank. We averaged all neural control-surface values in a square region over the two input variables x and ϕ. We assigned the average value to one of seven output fuzzy sets. Fig. 16 shows the resulting FAM bank and corresponding control surface generated by the neural control surface in Fig. 5(b). This new control surface resembles the original fuzzy control surface in Fig. 5 more than it resembles the neural control surface. Note the absence of a steady-state FAM rule in the FAM matrix in Fig. 5(a).

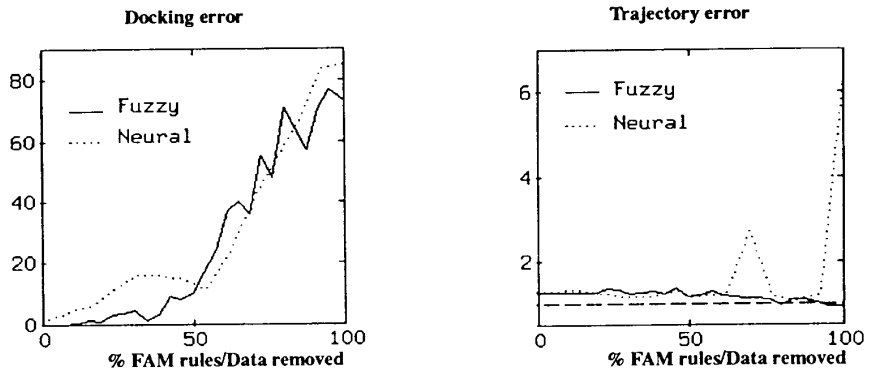
Fig. 17 compares the DCL-AFAM and BP-AFAM control surfaces with the fuzzy control surface in Fig. 5(a). Fig. 17 shows the absolute difference of the control surfaces. As expected, the DCL-AFAM system produced less absolute error than the BP-AFAM system produced.

Fig. 18 shows the docking errors of the DCL-AFAM and BP-AFAM control systems. The DCL-AFAM system produced less docking error than the BP-AFAM system produced for 100 arbitrary backing-up trials. The two AFAM systems generated similar backing-up trajectories. This suggests that black-box neural estimators can define the front end of FAM-structured systems. In principle we can use this technique to generate structured FAM rules for *any* neural application. We can then inspect and refine these rules and perhaps replace the original neural system with the tuned FAM system.

IV. Truck-and-Trailer Control Systems

A. Fuzzy Truck-and-Trailer Control System

We added a trailer to the truck system, as in the original Nguyen−Widrow model. Fig. 19 shows the simulated truck-and-trailer system. We added one more variable (cab angle, ϕ_c) to the three state variables of the trailerless truck. In this



(a) (b)

Fig. 24. Comparison of robustness of the two truck-and-trailer controllers: (a) docking error and (b) trajectory error.

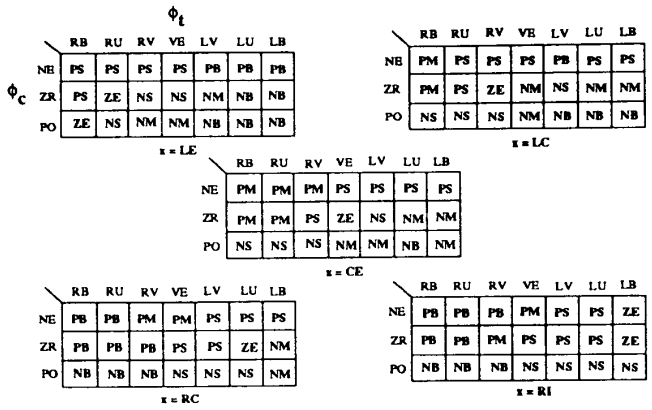


Fig. 25. DCL-estimated FAM bank for the AFAM truck-and-trailer system.

case a FAM rule takes the form

$$\text{IF } x = LE \quad \text{AND} \quad \phi_t = RB \quad \text{AND} \quad \phi_c = PO, \quad \text{THEN } \beta = NS.$$

The four state variables x, y, ϕ_t, and ϕ_c determined the position of the truck-and-trailer system in the plane. Fuzzy Fuzzy variable ϕ_t corresponded to ϕ for the trailerless truck. Fuzzy variable ϕ_c specified the relative cab angle with respect to the center line along the trailer (ϕ_c ranged from $-90°$ to $90°$). The extreme cab angles $90°$ and $-90°$ corresponded to two "jackknife" positions of the cab with respect to the trailer. A positive ϕ_c value indicated that the cab resided on the left-hand side of the trailer. A negative value indicated that it resided on the right-hand side. Fig. 19 shows a positive angle value of ϕ_c.

Fuzzy variables x, ϕ_t, and ϕ_c defined the input variables. Fuzzy variable β defined the output variable. β measured the angle that we needed to update the trailer at each iteration. We computed the steering-angle output θ with the following geometric relationship. With the output β value computed, the trailer position (x, y) moved to the new position (x', y'):

$$x' = x + r\cos(\phi_t + \beta) \tag{26}$$

$$y' = y + r\sin(\phi_t + \beta), \tag{27}$$

where r denotes a fixed backing distance. Then the joint of the cab and the trailer (u, v) moved to the new position (u', v'):

$$u' = x' - l\cos(\phi_t + \beta) \tag{28}$$

$$v' = y' - l\sin(\phi_t + \beta) \tag{29}$$

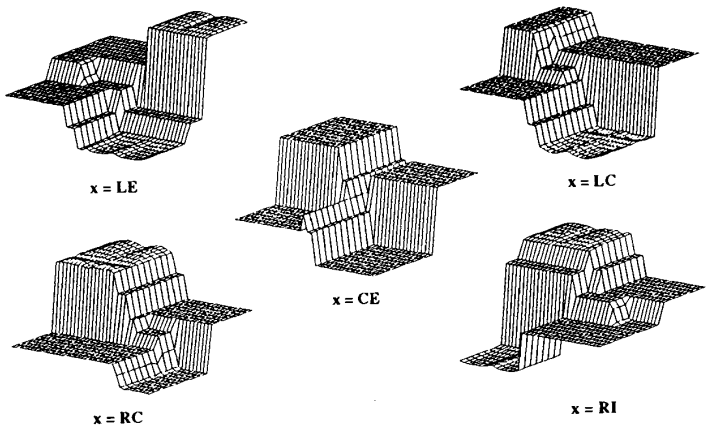
where l denotes the trailer length. We updated the directional vector $(\text{dir } U, \text{dir } V)$, which defined the cab angle, by

$$\text{dir } U' = \text{dir } U + \Delta u \tag{30}$$

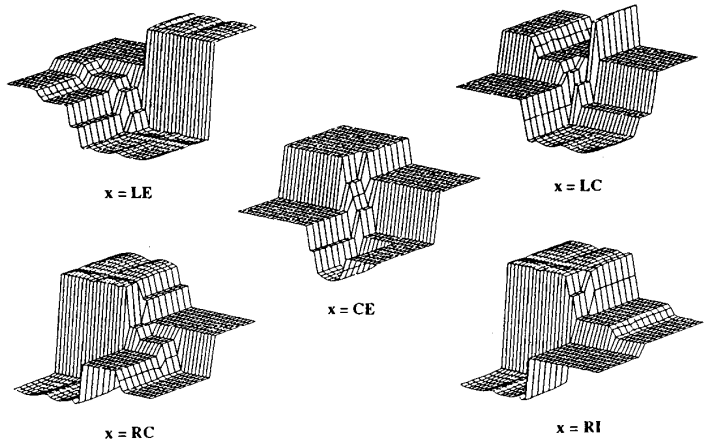
$$\text{dir } V' = \text{dir } V + \Delta v. \tag{31}$$

where $\Delta u = u' - u$, and $\Delta v = v' - v$. The new directional vector $(\text{dir } U', \text{dir } V')$ defines the new cab angle ϕ_c'. Then we obtain the steering angle value as $\theta = \phi_{c,h}' - \phi_{c,h}$, where $\phi_{c,h}$ denotes the cab angle with the horizontal. We chose the same fuzzy-set values and membership functions for β as we chose for θ. β ranged from $-30°$ to $30°$. We chose the fuzzy set values of ϕ_c as NE, ZR, and PO as in Fig. 20.

Fig. 21 displays the five FAM-rule matrices in the FAM bank of the fuzzy truck-and-trailer system. In Fig. 21 we fixed the fuzzy variable x as LE, LC, CE, RC, and RI. There were



(a)



(b)

Fig. 26. (a) Original control surfaces and (b) DCL-estimated control surfaces for the truck-and-trailer system.

735 ($7 \times 5 \times 7 \times 3$) possible FAM rules and only 105 actual FAM rules.

Fig. 22 shows typical backing-up trajectories of the fuzzy truck-and-trailer control system from different initial positions. The truck-and-trailer backed up in different directions, depending on the relative position of the cab with respect to the trailer. The fuzzy control systems successfully controlled the truck-and-trailer in jackknife positions.

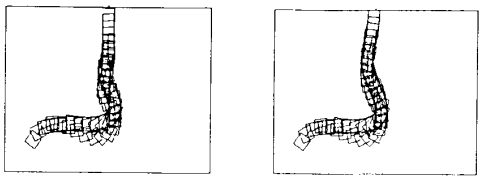
B. Neural Truck-and-Trailer Control System

We added the cab-angle variable ϕ_c as to the back-propagation-trained neural truck controller as an input. The controller network contained 24 hidden neurons with output variable β. The training samples consisted of five-dimensional space of the form $(x, y, \phi_t, \phi_c, \beta)$. We trained the controller network with 52 training samples from the fuzzy controller: 26 samples for the left half of the plane, 26 samples for the right half of the plane. We used (26)–(31) instead of the emulator network. Training required more than 200 000 iterations. Some training sequences did not converge. The BP-trained controller performed well except in a few cases. Fig. 23 shows typical backing-up trajectories of the BP truck-and-trailer control system from the same initial positions used in Fig. 22.

We performed the same robustness tests for the fuzzy and BP-trained truck-and-trailer controllers as in the trailerless truck case. Fig. 24 shows performance errors averaged over ten typical back-ups from ten different initial positions. These performance graphs resemble closely the performance graphs for the trailerless truck systems in Fig. 12.

C. Adaptive Fuzzy Truck-and-Trailer Control System

We generated 6250 truck-and-trailer data usign the original FAM system in Fig. 21. We backed up the truck-and-trailer from the same initial positions as in the trailerless truck case. The trailer angle, ϕ_t, ranged from $-60°$ to $240°$, and the cab angle, ϕ_c, assumed only the three values: $-45°$, $0°$, and $45°$. The training vectors $(x, \phi_t, \phi_c, \beta)$ defined points in the four-dimensional input–output product space. We nonuniformly partitioned the product space into FAM cells to allow narrower fuzzy-set values near the steady-state FAM cell.



(a) (b)

Fig. 27. Sample trajectory of (a) the original and (b) the DCL-AFAM system from the initial position $x = 30$, $y = 30$, $\phi_t = 10$, and $\phi_c = 45$.

We used DCL to train the AFAM truck-and-trailer controller. The total number of FAM cells equaled 735 ($7 \times 5 \times 7 \times 3$). We used 735 synaptic quantization vectors. The DCL algorithm classified the 6250 data into 105 FAM cells. Fig. 25 shows the estimated FAM bank by the DCL algorithm. Fig. 26 shows the original and DCL-estimated control surfaces for the fuzzy truck-and-trailer systems.

Fig. 27 shows the trajectories of the original FAM and the DCL-estimated AFAM truck-and-trailer controllers from the initial position $(x, y, \phi_t, \phi_c) = (30, 30, 10, 45)$. The original FAM and DCL-estimated AFAM systems exhibited comparable truck-and-trailer control performance except in a few cases, where the DCL-estimated AFAM trajectories were irregular.

V. Conclusion

We quickly engineered fuzzy systems to successfully back up a truck and truck-and-trailer system in a parking lot. We used only common-sense and error-nulling intuitions to generate sufficient banks of FAM rules. These systems performed well until we removed over 50% of the FAM rules. This extreme robustness suggests that, for many estimation and control problems, different fuzzy engineers can rapidly develop prototype fuzzy systems that perform similarly and well.

The speed with which the DCL clustering technique recovers the underlying FAM bank further suggests that we can similarly construct fuzzy systems for more complex, higher-dimensional problems. For these problems we may have access to only incomplete numerical input–output data. Pure neural-network or statistical-process-control approaches may generate systems with comparable performance. But these systems will involve far greater computational effort, will be more difficult to modify, and will not provide a structured representation of the system's throughput.

Our neural experiments suggest that whenever we model a system with a neural network, for little extra computational cost we can generate a set of structured FAM rules that approximates the neural system's behavior. We can then tune the fuzzy system by refining the FAM-rule bank with fuzzy-engineering rules of thumb and with further training data.

References

[1] P. J. Huber, *Robust Statistics*. New York: Wiley, 1981.

[2] S.-G. Kong, and B. Kosko, "Differential competitive learning for centroid estimation and phoneme recognition," *IEEE Trans. Neural Networks*, vol. 2, pp. 118–124, Jan. 1991.

[3] B. Kosko, "Fuzzy entropy and conditioning," *Inform. Sci.*, vol. 40, pp. 165–174, 1986.

[4] B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[5] B. Kosko, "Unsupervised learning in noise," *IEEE Trans. Neural Networks*, vol. 1, pp. 44–57, Mar. 1990.

[6] B. Kosko, "Stochastic competitive learning,"*IEEE Trans. Neural Networks*, vol. 2, pp. 522–529, Sept. 1991.

[7] D. Nguyen and B. Widrow, "The truck backer-upper: An example of self-learning in neural networks," in *Proc. Int. Joint Conf. Neural Networks (IJCNN-89)*, vol. II, June 1989, pp. 357–363.

Seong-Gon Kong received the B.S. and M.S. degrees in electrical engineering from Seoul National University, Seoul, Korea, in 1982 and 1987. He was then with the Electronics and Telecommunications Research Institute, Taejon, Korea. In 1991 he received the Ph.D. degree from the University of Southern California, Los Angeles.

He is currently an Assistant Professor of Electrical Engineering at Soongshil University, Seoul, Korea. His research interests include fuzzy systems, artificial neural networks, image processing, and data compression.

Bart Kosko (M'85) is an Assistant Professor in the Electrical Engineering Department of the University of Southern California.

He received bachelor degrees in philosophy and economics from the University of Southern California, the masters degree in applied mathematics from the University of California at San Diego, and the Ph.D. in electrical engineering from the University of California at Irvine.

Dr. Kosko is an elected member of the governing board of the International Neural Network Society, managing editor of Sprinter-Verlag's *Lecture Notes in Neural Computing* monograph series, and a former associate editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS, Neural Networks, the Journal of Mathematical Biology, and *Lecture Notes in Biomathematics*. He was program chairman of the 1987 and 1988 IEEE International Conferences on Neural networks, program cochairman of the 1990 International Joint Conference on Neural Networks (IJCNN-90), and program cochairman of the 1990 International Fuzzy Logic and Neural Networks Conference in Iizuka, Japan, and will be program cochairman of the IJCNN-93 in Portland, OR. Dr. Kosko is a USC Shell Oil Faculty Fellow and author of the Prentice-Hall textbook *Neural Networks and Fuzzy Systems* and editor of the Prentice-Hall volume *Neural Networks for Signal Processing*.