Paper

A Car-Following Model Based on a Self-Learning Neuro-Fuzzy Controller

Non-member Mohamed Anis Ben Amor (The University of Electro-Communications)
Member Watanabe Shigeyoshi (The University of Electro-Communications)

This paper is concerned with the development of a car-following model for traffic simulation. A new approach to emulate the driver's decision making mechanism is proposed. This approach uses the powerful techniques of fuzzy reasoning to deal with uncertainty and to smoothly blend behaviors induced by simultaneous goals, while being aware of the long history of equation based modeling. Moreover, an original algorithm is implemented in this system to imitate the driver's self-learning from his on-road driving experience and to mimic his maneuvers over the brake and the accelerator pedals which reflect his approximation of the appropriate acceleration for the actual situation.

Keywords: modeling, car-following, traffic simulation, fuzzy reasoning, neural networks, self-learning.

1. Introduction

Traffic congestion is a rising problem from which suffer many cities around the world. Many solutions that fall under the umbrella of Intelligent Transportation Systems (ITS) have been proposed to help solve this frustrating problem, including Advanced Traffic Management System (ATMS) for traffic assignment, Advanced Traveler Information System (ATIS) for dynamic route guidance and Automated Highway System (AHS).

Traffic network is a highly dynamic, complex and uncertain environment. A variety of approaches have been developed to modelize this system in an attempt to simulate and evaluate the effect of using ITS technologies before their costly implementation in real world(1).

Classically, in traffic modeling theory, these approaches can be categorized to two main levels: macroscopic and microscopic . Researchers who incorporate individual behavior of vehicles in their simulation models (microscopic), the main components of which are acceleration and lane changing, have achieved more realistic results compared to those who model traffic as fluid flow through a network (macroscopic)(2). Since the 1950s, researchers interested in simulating the traffic at the microscopic level, have extensively studied the modeling of the driver's acceleration behavior, commonly referred to as the car-following model; they have been focusing on equation based modeling (EBM). To see some examples, refer to(11),(3)∼(6).

These efforts have proved some successes in simulating a real-world traffic network. However, they do not faithfully emulate the complex and multi-ruled behavior of the driver in an uncertain and partially known environment such as traffic roads, as it was pointed out by S.Kikuchi et al.(7). In fact, the driver's behavior is influenced by a vague perception of the surrounding environment (approximate velocity, headway, etc.) and his decision making mechanism is driven by fuzzy rules. To overcome these deficiencies, new ideas have been proposed throughout recent years, by using the powerful inference mechanism in fuzzy logic which allows a human-like reasoning and knowledge base to be directly implemented in the model. The difference between the two approaches is that, in EBM, the target system is formulated as an accurate mathematical model, whereas in fuzzy based modeling, an experienced human operator of the system is modeled. To see some examples, refer to(7)∼(10).

This paper is concerned with the development of a car-following model for traffic simulation environment. A novel approach is proposed to imitate the driver's behavior and decision making mechanism. While considering the equation based modeling, the proposed approach takes advantage of the techniques used in fuzzy set theory to manipulate uncertainty and to smoothly blend behaviors induced by simultaneous goals.
Many developed models in the literature of traffic simulation capture variety between drivers and classify them as beginners or experts using some predefined parameters. In this paper however, a learning algorithm is developed and incorporated in the system to emulate as 'naturally' as possible the learning process of the driver from his on-road driving experience. This algorithm allows also to mimic the driver's manipulation of the brake and the accelerator pedals, which reflects his approximation of the appropriate acceleration for the actual situation.

The proposed fuzzy logic controller is detailed in *section 2*, the architecture of the system is sketched in *section 3*, and description of the learning algorithm is given in *section 4*. In *section 5*, simulation results are presented along with their related discussions. The conclusion includes a short description of the ongoing work.

2. The Fuzzy Logic Controller

Our car-following model has been originally inspired by the model proposed by Qi Yang(4), which uses an equation based approach to calculate the acceleration for different regimes: 1) Free-flowing regime: the headway is larger than a defined threshold h^{upper}, there is no interaction with the front vehicle ; 2) Emergency regime: the headway is smaller than a defined threshold h^{lower}, the driver decelerates to avoid collision with the leading vehicle and keeps a safe headway; and finally, 3) Car-following regime: the headway is between h^{upper} and h^{lower}.

Unlike the original model, the proposed model deals with these regimes in a fuzzy context. This section is a detailed description of the design of a fuzzy logic controller (FLC) for these 'fuzzy regimes'. The FLC consists of three inputs, one output and some 20 rules. Notation in the following paragraphs uses the index $_l$ for Leading vehicle's variables and $_f$ for Following vehicle's.

2.1 Input variables

Speed (V_f): Velocity of the following (subject) vehicle.

Linguistic terms: { slow, medium, fast, less (than) des(ired), desired, more (than) des(ired) }. Membership functions (MFs) are generalized bell functions.

The desired speed is mainly dependent on the traffic condition, driver skills and trip motivation. Its MF is defined mainly by its position, i.e., its center c. A fuzzy inference system (FIS) is used to compute c.

First, a normalized score in the interval [0,1] is given to the traffic condition as follows:

$$TC = \frac{Vs}{5}(1 - \frac{Td}{180})$$

where: Td is the traffic density, [0,180] Veh/Km/Lane. Vs is the visibility, [0,5] Km, measured with no obstacle ahead (e.g. car or building).

TC is then fuzzified using the terms "good", "medium" and "bad" with MFs centered respectively at 0, 0.5 and 1.

Rules of this fuzzy inference system (FIS) are:

(traffic condition==good)⇒(des_speed=fast)
(traffic condition==medium)⇒(des_speed=medium)
(traffic condition==bad)⇒(des_speed=slow)

Where "fast", "medium" and "slow" have the same MFs defined for speed variable.

Finally, $c = COA(des\_speed) + k$, where COA stands for centroid of area and k is a relatively small random number used to capture the variety between drivers in trip motivation.

The *decision curve for desired speed*, the overall input-output curve of this system, is given in Fig.1.

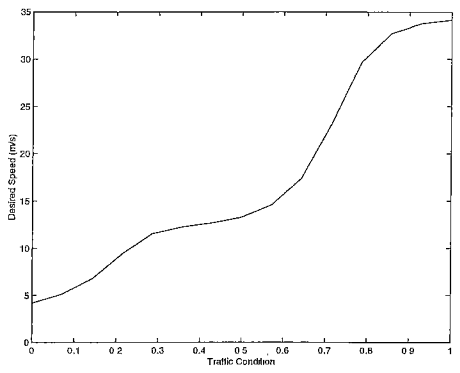


Fig. 1. Decision curve for desired speed

Relative Speed (R): Relative speed of the leading vehicle, $R = V_l - V_f$.

Linguistic terms: { neg(ative), zero, pos(itive) }. MFs are trapezoidal functions.

The relative speed perceived by the driver is a function of the headway, i.e., when the driver is too close to the front vehicle, "zero" tends to be a singleton [0], while "neg" and "pos" tend to be crisp sets, [-50,0] and [0,50] respectively. For a larger headway, the membership function "zero" tend to have larger core[†] and larger slope. "neg" and "pos" MFs move accordingly satisfying the condition of $\epsilon - completeness$ with $\epsilon = 0.5$.

Headway (H): Distance to the leading vehicle.

Linguistic terms: { small, medium, large, not_safe }. MFs of the three first terms are generalized bell functions.

The term "not_safe" corresponds to the minimum distance such that, if the front vehicle brakes with its maximum deceleration rate (MDR_l) for any reason and the following vehicle does so to avoid collision, the following vehicle will stop at an arbitrary stopping distance $Stop$ to the front vehicle, (e.g. $Stop = 2m$).The minimum safe distance is calculated using the following equation:

$$S = \frac{V_l^2}{2 * MDR_l} - \frac{V_f^2}{2 * MDR_f} + Stop$$

The "not_safe" MF is a trapezoidal function with parameters {a,b,c,d}, where: $a = b = 0$, $c = S$, and d has an arbitrary value. The distance $d - c$ will be called *safe distance offset*.

A fourth variable, the leading vehicle's acceleration (Al), is omitted: it takes the term "any" in all the rules. Use of this variable will be explained in the next section.

2.2 Output variable

Acceleration (A_f): Acceleration of the following vehicle.

Linguistic terms: { (brake) hard, normal (deceleration), neg(ative) med(ium), zero, pos(itive) med(ium), max(imum) }.

- When "max", the acceleration is equal to the maximum acceleration rate (MAR) which depends on the actual speed, see Tab.1.

Table 1. Acceleration Rates[††]

Speed(m/s)	<6	6-12	12-18	18-24	≥24
$MAR(m/s^2)$	3	2.4	1.7	1.2	1.2
$MDR(m/s^2)$	-3	-2.89	-2.74	-2.59	-2.43
$NDR(m/s^2)$	-2.57	-2.21	-1.58	-1.58	-1.58

- "hard" is the acceleration when the driver decelerates to avoid collision with the front vehicle or extends his headway to a safe range in emergency

[†]The core of a fuzzy set A is defined as: $core(A) = \{x|\mu_A(x) = 1\}$.
[††]The same values in [4] are used in this paper.

regime. Mathematically speaking, this acceleration is a function of $\{A_l, R, H\}$ and limited to the MDR (see Tab.1) as shown below:

$$A_f = max\{MDR, A_l - \frac{R^2}{2H}\} \cdots\cdots\cdots\cdots (1)$$

- When "normal", the acceleration is equal to the normal deceleration rate (NDR), see Tab.1. It corresponds to the case when the driver decelerates to keep his desired speed in free-flowing regime.
- The "pos_med" and "neg_med" terms correspond to the car-following regime; the acceleration is calculated based on Herman's general car-following model[11]:

$$A_f = \alpha^{\pm} \frac{V_f^{\beta^{\pm}}}{H^{\gamma^{\pm}}} R \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (2)$$

Where $\alpha^{\pm}$, $\beta^{\pm}$ and $\gamma^{\pm}$ are model parameters. $\alpha^{+}, \beta^{+}, \gamma^{+}$ are used for accelerating ($V_f \leq V_l$), and α^{-}, β^{-}, γ^{-} for decelerating ($V_f > V_l$) cases. Default values for these parameters are taken from[12], see Tab.2.

Table 2. Herman Model Parameters

	α	β	γ
acceleration	2.15	-1.67	-0.89
deceleration	1.55	1.08	1.65

- In case of "zero", a small random number is assigned to the output variable.

The output terms correspond either to constants or functions of the input variables; the overall output is computed using the weighted average method:

$$O = \frac{\sum_r w_r o_r}{\sum_r w_r}$$

where w_r and o_r are the firing strength and the output of the r^{th} rule, respectively. The proposed fuzzy inference system is considered as of Sugeno type[13].

2.3 Rule Base Formulation of the rules is based on a real human expertise on driving. The rule base can be divided into three sets corresponding to the three regimes.

Free-flowing regime is governed by the three following rules:

(speed==less_des)&(headway==large)⇒(acc=max)
(speed==desired)&(headway==large)⇒(acc=zero)
(speed==more_des)&(headway==large)⇒(acc=normal)

In car-following regime, the acceleration is either "neg_med" or "pos_med" depending on the relative speed. Here is a sample of this rule base subset:

(speed==medium)&(rel==neg)&(headway==medium) ⇒(acc=neg_med)
(speed==medium)&(rel==pos)&(headway==medium) ⇒(acc=pos_med)

Emergency regime is translated into a single and unique rule:

(headway==not_safe)⇒(acc=hard)

3. System Architecture

A simplified diagram of the system architecture is shown in Fig.2.

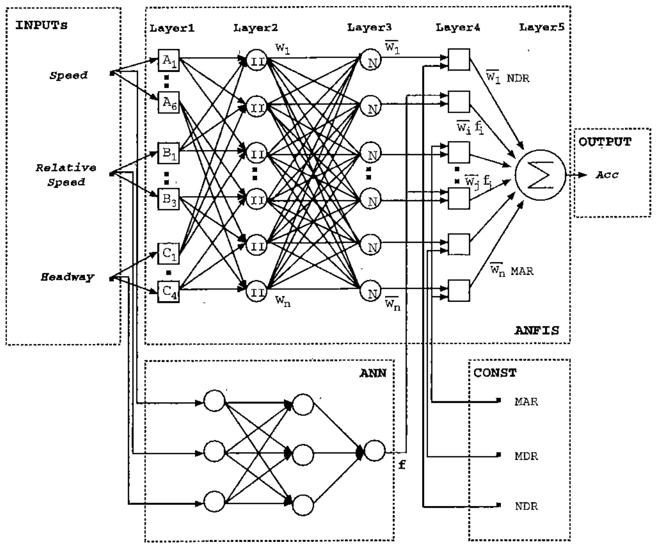


Fig. 2. System flowchart

A functionally equivalent adaptive network of the Sugeno fuzzy inference system is shown in the box titled ANFIS[14]. Here is a brief description of each layer in the ANFIS:

Layer 1 The node function of a node i in this layer is:

$$O_{1,i} = \mu_{A_i}(x)$$

Where x is the input (e.g. speed), A_i is a linguistic term (e.g. "slow", "fast", etc.) associated with this variable and μ_{A_i} is its membership function. In other words, $O_{1,i}$ is the membership grade of x in A_i. Parameters of this layer are the parameters of the membership function $\mu_{A_i}(x)$. They are referred to as *premise parameters.*

Nodes corresponding to "slow", "medium", "fast" for speed variable and "small", "medium", "large", "not_safe" for headway variable are the only adaptive nodes in this layer.

Layer 2 Every node in this layer is a fixed node where a T-Norm operator, an algebraic product in our system, is applied to the incoming signals.

$$O_{2,i} = w_i = \mu_{A_i}(x)\mu_{B_i}(y)\mu_{C_i}(z)$$

In other words, each node output represents the firing strength of a rule.

Layer 3 Every node in this layer is a fixed node. The ith node calculates the ratio of the ith rule's firing strength to the sum of all the rules' firing strengths.

$$O_{3,i} = \bar{w}_i = \frac{w_i}{\sum_i^n w_i}$$

Outputs of this layer are called *normalized firing strengths.*

Layer 4 Every node i in this layer is an adaptive node with a node function:

$$O_{4,i} = \bar{w}_i c_i$$

c_i is the value of the output of the ith rule, (in Fig.2, it is equivalent to MAR, MDR, NDR, or f_i).

Typically, for a first-order Sugeno FIS, $c_i = f_i(x,y,z) = p_i x + q_i y + r_i z + s_i$. Parameters of this layer are the parameters of the function $f_i = \{p_i, q_i, r_i, s_i\}$, they are referred to as *consequent parameters*.In our model, f_i is the function of the corresponding neural network box and its weight matrix is the consequent parameter.

Note that the acceleration of the leading vehicle (A_l) is not a direct input to the fuzzy system but rather an input to the neural network of the corresponding rule.

Layer 5 The single node in this layer is a fixed node that computes the overall output as a summation of all incoming signals.

$$O_{5,1} = \sum_i \bar{w}_i c_i = \frac{\sum_i w_i c_i}{\sum_i w_i}$$

This representation helps to understand how to implement a learning algorithm into this system for the premise and the consequent parameters. This issue, together with the function of the ANN box will be explained and discussed in the next section.

4. Learning Algorithm

The main motivation behind the development of this algorithm is to emulate the learning of the driver from his on-road driving experience. It is based on the idea that drivers 'naturally' enhance their skills by driving on the road. The algorithm is hybrid, it is composed of two routines to train simultaneously premise and consequent parts of the fuzzy controller, as shown in Fig.3.

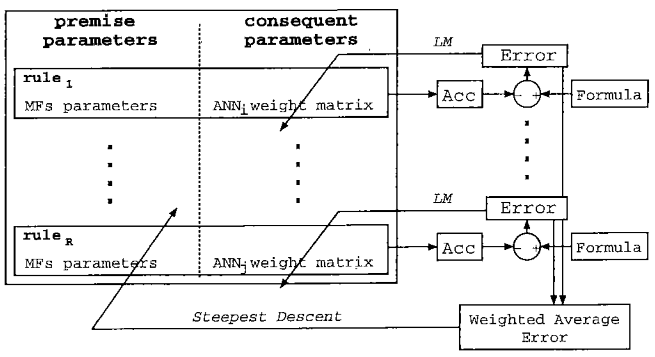


Fig. 3. Learning Algorithm

4.1 Consequent Learning As cited in a previous section (see 2.2), the output of the system is a function of its inputs. More specifically, when the output is "hard", "pos_med" or "neg_med", computation of the acceleration is based on Equations 1 and 2. In real world, behavior of a driver is, indeed, a reflextion of his approximation of these functions. The actual learning routine is built upon this idea.

Artificial neural network (ANN) are well known for their ability to imitate humans in learning from experience, in other words, given a set of input/output data, they can perform an approximation (with certain error) of the output even if the input is not exactly the same as the already learned one.

The output of a rule that has in the consequent part "hard", "neg_med" or "pos_med" will be computed by a corresponding feed-forward neural network (see Fig.2). This type of rules is called *neural rule*.

To train these neural networks, a fast convergence algorithm in which the performance function is always reduced at each iteration of the training process is needed. The Levenberg-Marquardt algorithm (LM) satisfies these two conditions; It is known to be the fastest method for training moderate-sized feed-forward neural networks(15). The objective function to be minimized is the mean squared error's (mse).

4.1.1 Pre-Training (batch) This training phase may correspond to the 'driving school' phase that every driver must pass. It can be addressed as follows:

Start- Initialization of the neural networks: The ANN starts with matrix of weights with random numbers, which may correspond to a very beginner driver.

Step 1- Training Data: prepare a set for every ANN. (for details, see next section Simulation/Set up).

Step 2- Training: Perform one training iteration (epoch) by presenting to the ANN the whole set of training data.

Step 3- Verify stopping condition: If the objective function has reached a defined minimum value, $mse \le MSE_0$, then stop training (go to *End*), otherwise go to *Step 2*.

End- Stopping condition is reached: From this point, the skills of the driver are considered acceptable and he is allowed to drive on the road.

4.1.2 On-Road Training (on-line) Now the driver is on a 'real' road. Continuously over time (every time step in the simulation), The following steps are applied:

Start- $MSE = MSE_0$.

Step 1- Compute the error: $e_r = t_r - o_r$, where t_r is the target output and o_r is the actual output of the network.

Step 2- Compare the error with the performance of the network:

```
If $e_r^2 > MSE$, then
  premise_update = TRUE.
else quit.
```

Step 3- Extend the training set: actual (input/target output) pair is added to the list of training data.

Step 4- Training: Perform one training iteration.

Step 5- Stopping conditions:

```
1-If $mse \le MSE$, then
    if $mse > MIN$, then
      $MSE = mse$
    else $MSE = MIN$
    quit.
2-If a max number of epochs is reached, then
    extract the actual (input/target output) pair
```

from the training list.
premise update = FALSE
quit.

If none is verified, go to *Step 4*.

if premise_update == TRUE then premise learning (see next paragraph).

MIN is an arbitrary minimum error value. The constraint in condition 1 will ensure that the ANN is trained only for an error measure bigger than MIN, and $\forall n, MIN \leq MSE_n \leq MSE_0$.

Fig.4 is an illustration of the error measure during Pre and On-road training phases. The major part of error reduction is performed in batch training phase. The on-road training is performed occasionally (see step 2); it is infact the same as the batch training but with an extended training set. Convergence of the network to a new MSE is reached within fewer epochs.

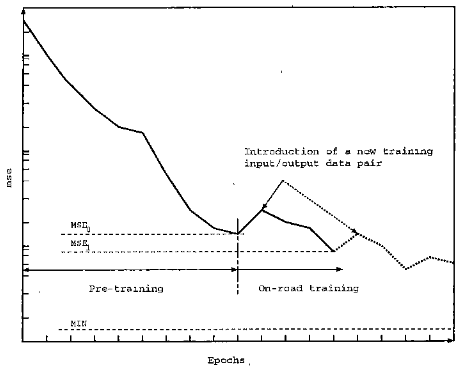


Fig. 4. ANN Training Process

4.2 Premise Learning An experienced driver and a beginner driver may have a different understanding of 'headway is small', 'speed is fast', 'headway is not safe', etc. In our system, this implies that the MFs of the input variables may change their positions and even their shapes to reflect the driver's view of the driving state.

For this matter, a learning algorithm that goes in parallel with the above described one is implemented in the ANFIS. The parameters are updated by Steepest Descent method (SD) based on the backpropagation of the overall error in the driver's action, which is the weighted average of all the rules'output errors:

$$T - O = \frac{\sum_r w_r(t_r - o_r)}{\sum_r w_r}$$

The error measure is the squared error:

$$E = \frac{1}{2}(T - O)^2$$

Let a be a parameter in a rule r, the method to update a for this rule is as follows:

$$\begin{aligned}\Delta_r a &= -\eta_a \frac{\partial E}{\partial a} \\ &= -\eta_a \frac{\partial E}{\partial w_r}\frac{\partial w_r}{\partial a} \\ &= -\eta_a (T - O)\frac{(t_r - o_r) - (T - O)}{\sum w}\frac{\partial w_r}{\partial a}\end{aligned}$$

Where η_a is the learning rate of the parameter a, T and O are the overall target output and the overall actual output, respectively. t_r, o_r, and w_r are the target output, the actual output, and the firing strength of the r^{th} rule, respectively.

4.3 Remarks The system is *self supervised* by the mathematical model instead of real-world data which could be very erroneous or heavily influenced by some facts that are hard to catch. Preparation of training data is detailed in next section.

The consequent learning has a major and direct effect on premise learning, since the error measure used to update the premise part is the weighted average of the errors engendered by the neural rules (see Fig.3). The premise learning has no effect on consequent learning, consequently, the *convergence* of the whole learning algorithm is driven by the convergence of the neural networks in consequent learning to the defined minimum error MIN.

The on-road training procedure will prevent the ANN from *over-fitting* since it performs *generalization* in a natural way: data that represents erroneous driving maneuvers, in other words, occasionally 'mistakes' of the driver, are added to the training set as to enrich the 'driver's experience'.

To maintain the *interpretability* of the trained MFs, some constraints are set for the parameters update procedure:

- The update is performed for all the parameters except the center parameters for "slow", "fast", "small" and "large" MFs.
- The centers of all the MFs must keep their order over the universe of discourse. This will prevent the MFs from passing each other.
- The update is allowed only for a specific direction to achieve a comprehensive modification of the MFs.

5. Simulation

5.1 Set up The simulator uses the Object Oriented Programming paradigm provided in Matlab.

Training Data: For each ANN, a list of (input/target output) pairs is prepared. An input is a vector whose elements are samples of the corresponding input variables. The target output is calculated using this vector with the appropriate equation, (e.g.: in emergency regime, the input is $[A_l, R, H]^T$, the target is calculated with Eq. 1). Sampling is done uniformly over the compact set of each variable. Sampling steps are : 1 for A_l, 10 for V_l, 10 for H, and 10 for R.

Training of ANNs during the simulation is performed in two phases as explained earlier.

Phase 1: Pre-training.
ANNs are trained in batch mode using LM algorithm until an error measure $MSE_0 = 0.1$ is reached. Generally, the ANNs converge to this value within 10 to 15 epochs.

Phase 2: On-road training.

Generally, convergence to a new MSE is reached within 1 to 5 epochs. The minimum MSE is chosen to be $MIN = 0.01$.

The time step is chosen to be $ts = 0.05s$. Computation of position, speed and acceleration is made in floating point, thus, the model is continuous.

5.2 Results and Discussions

5.2.1 System Evaluation by Example

Evaluation of the system is done by simulating different driving situations. Initial values of the variables in this simulation are chosen as to reproduce the same scenarios (specially 1 and 2) used by S.Huang et.al in [10] in purpose to encourage the reader to compare the results. The curves shown in Figs.5, 6 and 7 represent the behavior of the system (driver/vehicle) in terms of acceleration, speed and headway when starting from the initial conditions, as detailed in the following paragraphs.

First scenario

Initially, the subject vehicle is running with a speed of $V_f = 20m/s$. A vehicle changes lane in front of it at a distance of 10 m with a relative speed $R = +6m/s$. The reaction of the driver to this situation is given in Fig.5

The driver keeps following the front vehicle while being aware to not violate the safe distance. When the headway begins to be 'large enough', he gradually accelerates to reach his desired speed. When the desired speed is reached, the driver tries to keep it.

Second scenario

The subject vehicle is running with a speed of $V_f = 20m/s$. A vehicle changes lane in front of it at a distance of 30 m and keeps running at a speed $V_l = 10m/s$.

At first, the driver acts in emergency regime as to avoid collision with the cutting vehicle, he decelerates until a safe combination of relative speed/headway is reached. Then, he keeps following the front vehicle in car-following regime, see Fig.6.

Third scenario

The subject vehicle is stopped. A vehicle in front of it is stopped at a distance of 120 m.

The driver starts with increasing his speed, having in mind the purpose of reaching the desired speed. He realizes after a certain time that the headway does not allow him to do so, he then decelerates to stop at an arbitrary distance before colliding with the front vehicle, see Fig.7.

Small oscillations observed in the acceleration curves are due to the effect of ANNs. As the ANNs learn during the on-road driving, these oscillations will gradually disappear. They reflect the driver's approximation of the appropriate acceleration for the actual situation by manipulating the brake and the accelerator pedals.

5.2.2 Convergence of the Learning Process

The convergence of the learning process is proved by an experiment. The experiment consists of repeating the third driving scenario for 10 times.

In Fig.8, the root mean squared error of the overall driving time is decreasing at each run and converging to a certain minimum value.

In Fig.9, NBH, NBP, and NBN are the numbers of

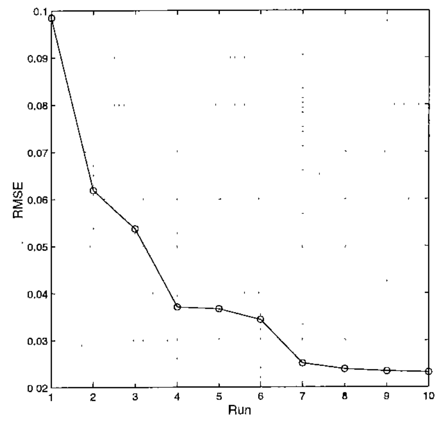


Fig. 8. Root Mean Squared Error

added pairs to the training data for ANNs corresponding to "hard", "pos_med" and "neg_med" respectively. These numbers gradually converge to 0.

It is concluded from the above two observations, that the learning process converges and the MFs will have final forms.

5.2.3 Effect of Learning to the Membership Functions

An example of the effect of this learning process to the MFs is given from the consecutive executions of the three previous scenarios.

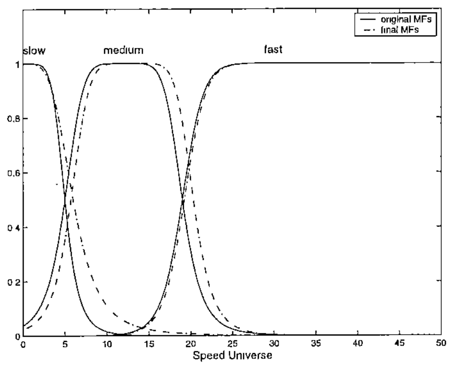


Fig. 10. Learned *Speed* MFs

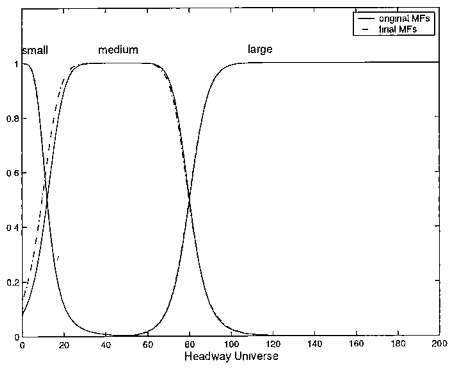


Fig. 11. Learned *Headway* MFs

Intuitively, a beginner driver is more careful than an expert because he has more fear to make accidents. He

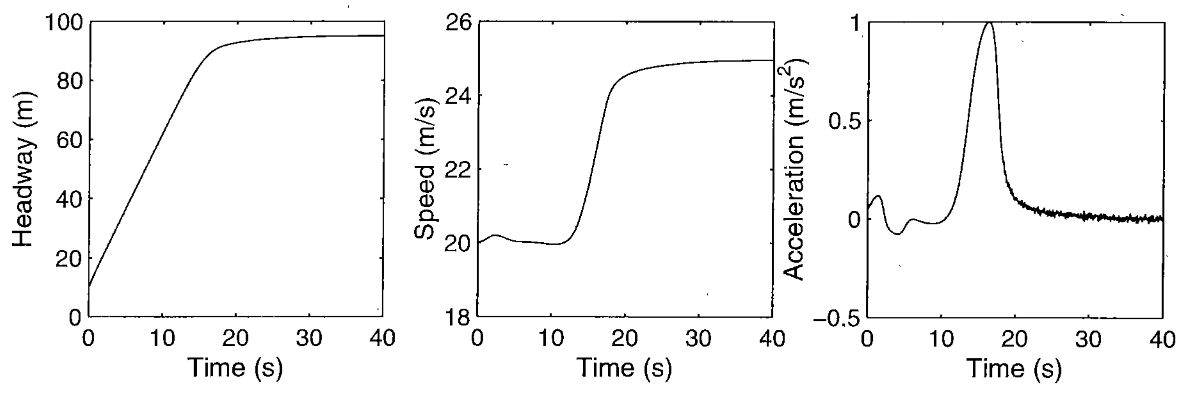


Fig. 5. 1st Scenario

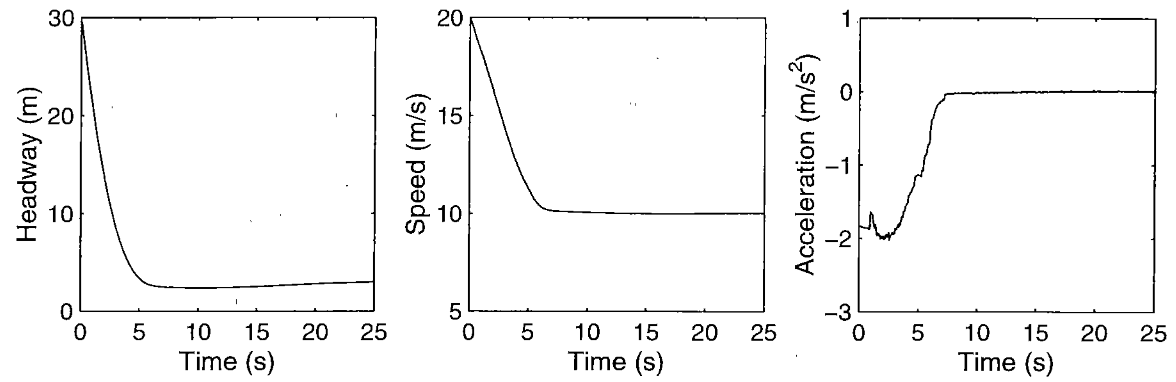


Fig. 6. 2nd Scenario

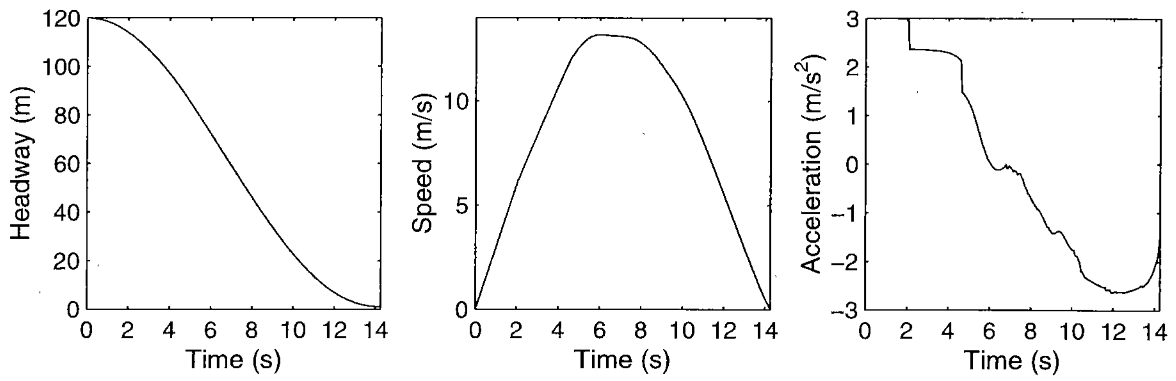


Fig. 7. 3rd Scenario

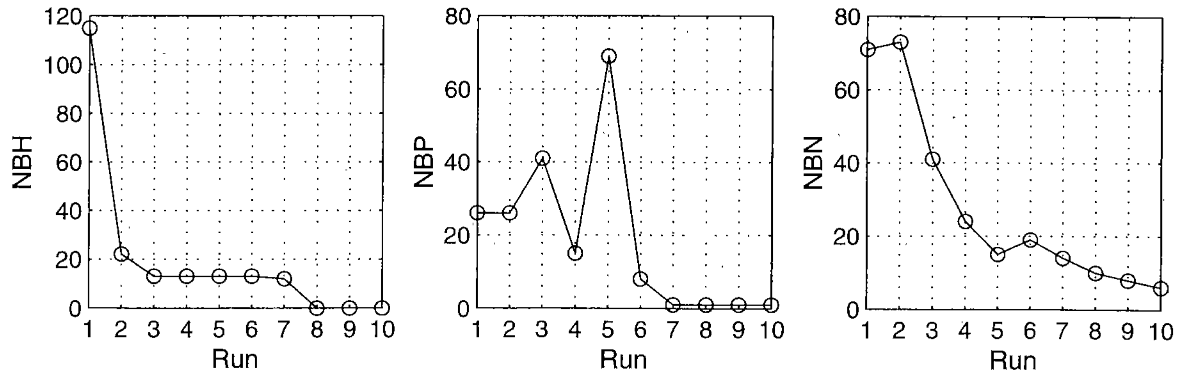


Fig. 9. Number of Added Pairs to the Training Sets

tries always to keep a larger headway and drive in a moderate speed comparing to an expert driver who is more confident in his skills and ability to make good decision at the right time.

Figures 10 and 11 show that, a beginner driver may consider a given speed as medium, while the same speed is considered by an experienced driver as still slow; This explains why the speed MFs shift to the right. Note that this modification will affect the driver's desired speed: since the speed MFs are shifting to the right, the desired speed will be increased accordingly (see paragraph *Input variables: Speed*).

A beginner driver may consider a given headway small while the same headway is considered by an expert as still medium; This explains why the headway MFs shift to the left.

The safe distance offset will be gradually decreased as the driver has better approximation of the distance needed to smoothly brake without colliding with a braking vehicle in front of it.

6. Conclusion

This paper introduced a mechanism based on a combination of two major Artificial Intelligence branches, Fuzzy Logic and Neural Networks, to deal with the driver's decision making on acceleration.

While fuzzy reasoning deals with uncertainty in a driving environment, neural networks play the role of local experts in this system; they allow the emulation of the driver's learning process from his own experience while driving on the road, and the imitation of his manipulation of the brake and the accelerator pedals reflecting his approximation of the appropriate acceleration for the actual situation.

Results obtained from the simulation of different driving scenarios, are very comparable to those of[10], but emphasizing better human learning and approximation. Their clear aspects and understandable meanings are very encouraging for implementation in a traffic simulation environment.

The proposed system is very flexible. It allows easy formulation of the rule base, shapes of fuzzy sets can be calibrated by different methods (e.g. that of[8]), precision of the ANNs can be freely chosen as to have an appropriate approximation.

A natural extension to this work is the development of other driving behaviors, principally, the lane changing model, where similar techniques can be used to decide the appropriate steering.

(Manuscript received December 11, 2000, revised June 22, 2001)

References

(1) Matti Pursula. "Simulation of Traffic Systems - An Overview". Journal of Geographic Information and Decision Analysis, Vol.3, No.1, 1999, pp. 1-8.

(2) K.R.Howard. "Unjamming Traffic with Computers". Scientific American, Oct.1997.

(3) H.Raza and P.Ioannou. "Vehicle Following Control Design for Automated Highway Systems". IEEE, Control Systems, Dec.1996.

(4) Qi Yang. "A Simulation Laboratory for Evaluation of Dynamic Traffic Management Systems". Ph.D. dissertation, Massachussetts Institute of Technology, Jun.1997.

(5) Anthony Thedore and Charles Michael Johnston. "A Real-Time Traffic Simulation System". IEEE Transaction on Vehicular Technology, Vol.47, No.1, Feb.1998.

(6) Kazi Iftekhar Ahmed. "Modeling Drivers' Acceleration and Lane Changing Behavior". Ph.D. dissertation, Massachussetts Institute of Technology, Feb.1999.

(7) S.Kikuchi and P.Chakroborty. "Car-Following Model Based on Fuzzy Inference System". National Research Council, Washington DC, Transportation Research Record 1365 (TRB), 1992, pp. 82-91.

(8) J.Wu, M.Brackstone, and M.McDonald, "Fuzzy Sets and Systems for a Motorway Traffic Simulation Model". The International Journal of Fuzzy Sets and Systems, Vol.116, No.1, 2000, pp.65-76.

(9) Philippe Garnier and Thierry Fraichard. "A Fuzzy Motion Controller for Car-Like Vehicle". Institut National de Recherche en Informatique et Automatique (France), Research Report No.3200, Jun.1997.

(10) Sunan Huang and Wei Ren. "Use of Neural Fuzzy Networks with Mixed Genetic/Gradient Algorithm in Automated Vehicle Control". IEEE Transactions on Industrial Electronics, Vol.46, No.6, Dec.1999.

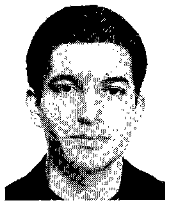
(11) R.Herman, E.W.Montroll, R.Potts, and R.W.Rothery. "Traffic Dynamics: Analysis of Stability in Car-Following". Operation Research, Vol.1 (7):-106, 1959.

(12) H.Subramanian. "Estimation of a Car-Following Model for Freeway Simulation". Massachussetts Institute of Technology, Cambridge, MA, 1996.

(13) T.Takagi and M.Sugeno. "Fuzzy Identification of Systems and its Application to Modeling and Control". IEEE Trans. Syst.,Man,Cybern., Vol. SMC-15, no 1, Jan. 1985, pp. 116-132.

(14) Jyh-Shing Roger Jang, Chuen-Tsai Sun, Eiji Mizutani. "Neuro-Fuzzy and Soft Computing: a Computational Approach to Learning and Machine Intelligence". Prentice Hall, (MATLAB curriculum series), 1997.

(15) Hagan, M.T., and M.Menhaj. "Training Feedforward Networks with the Marquardt Algorithm" IEEE Transactions on Neural Networks, Vol.5, No.6, 1994, pp.989-993.



Mohamed Anis Ben Amor (Non-member) was born on 1st of January 1973 in Tunis, Tunisia, and received B.S. degree from the Ecole Superieure des Postes et des Telecommunications de Tunis in 1995. He received M.E. degree from The University of Electro-Communications, Tokyo, Japan, in 1999. He is actually a Ph.D. candidate in the same university. His research interests include traffic simulation, massive parallel computation, artificial intelligence and multi-agent systems.

Shigeyoshi Watanabe (Member) was born on 19th of June, 1944 in Fukuoka and received the B.S., M.S. and Ph.D degrees in Electrical Engineering from Tokyo Institute of Technology, Japan in 1968, 1970 and 1973, respectively. From 1973 to 1991, he was a member of the faculty at Gunma University. Since 1991 he has been a member of the faculty at the University of Electro-Communications, where he is currently a professor at the department of Information and Communications Engineering. From 1978 to 1979 and from 1983 to 1984 he was a visiting professor at the Energy Systems Research Center, the University of Texas at Arlington, U.S.A.. His research interests include Massively Parallel Computation and Artificial Intelligence. He is currently the member of IEEE, IEICE, IEE, IPSJ and JSAI.