



ELSEVIER

Fuzzy Sets and Systems 125 (2002) 93–104

FUZZY
sets and systems

www.elsevier.com/locate/fss

A design of CMAC-based fuzzy logic controller with fast learning and accurate approximation [☆]

Daijin Kim ^{*}

Department of Computer Science and Engineering, POSTECH, San 31, Hyoja Dong, Nam-Gu, Pohang, 790-784, South Korea

Received 22 July 1998; received in revised form 29 February 2000; accepted 19 June 2000

Abstract

This paper proposes a CMAC-based fuzzy logic controller (FLC) with a fast learning capability and an accurate approximation ability. The proposed CMAC-based FLC has the fast learning capability because it pursues the local generalization and only a small number of activated units in the network are participated in the forward and backward computation. It also produces an accurate input–output approximation ability, because it adjusts the MF's model parameters of the input and output variables simultaneously and it considers both centers and widths of output membership functions to compute a crisp defuzzified value. Application to the truck backer-upper control problem of the proposed CMAC-based FLC is presented. Simulation results validate the fast learning and the accurate approximation of the proposed CMAC-based FLC. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Fuzzy logic controller; Cerebellar model articulation controller; Backpropagation learning; Truck backer-upper control

1. Introduction

Fuzzy logic controllers (FLCs) have been widely applied to both consumer products and industrial process controls. In particular, FLCs are very effective techniques for complicated and imprecise processes, for which either no mathematical model exists or the mathematical model is severely non-linear. They can easily approximate what human experts perform well under such ill-defined environments [4].

A typical configuration consists of four principal units: (1) fuzzifier which converts a crisp input to a fuzzy term set; (2) fuzzy rule base which stores fuzzy rules describing how the fuzzy system performs; (3) fuzzy inference engine which performs an approximate reasoning by associating input variables with fuzzy rules and

[☆] This paper is supported in part by the grant of KOSEF (1992-2-30200-047-3) and in part by the Ministry of Education of Korea through its BK21 program.

^{*} Tel.: +82-562-279-2249; fax: +82-562-279-2299.

E-mail address: dkim@postech.ac.kr (D. Kim).

(4) defuzzifier which converts the FLC's fuzzy output to a crisp value for the actual system input over the target.

In the conventional design method of FLC, the designers determine the membership functions and the fuzzy rules by their behavioral knowledge or subjective judgement. So, the control performance of the resultant FLC might not be an optimal one. Recently, FLC designers explore to design the FLC to have the capability of learning from examples by itself. One typical example of this direction is, to combine the network structure and learning capability of neural networks into the FLC.

Lin and Lee [3] proposed a general neural network (connectionist) model for FLC that could be constructed from training examples by the technique of learning from examples. Their connectionist structure could be trained to develop fuzzy logic rules and find optimal input/output membership functions. Their learning procedures consisted of two sequential stages: the first stage performed the unsupervised self-organizing learning that determined the initial parameters of membership functions and fuzzy logic rules, and the next stage performed the supervised learning that determined the fine-tuned parameters of membership functions. Also, the connectionist structure avoids the rule matching of the inference process. However, their approach requires very long training time in the sequential machine because all units are involved in the MLP's backpropagation learning. Jang [7] proposed the adaptive network-based fuzzy inference system (ANFIS) that was a fuzzy inference system implemented in the framework of adaptive networks. It could be used as a universal approximator that could model any nonlinear functions within a tolerable amount of approximation error. However, the system also required a lot of training time because it had very complex hybrid backpropagation learning procedures such as the least-squares estimate method in the forward pass and the gradient descent method in the backward pass.

Differently from the global weight adjustment of the backpropagation neural network, cerebellar model articulation controller (CMAC) model, proposed by Albus [1], has been attracted due to its local weight adjustment that exhibits faster learning speed and easier hardware implementation. After Albus, many related researches have been reported. Wong and Sideris [15] proved that the CMAC's learning could be always convergent within a tolerable precision. Moody [8] proposed the multiresolution CMAC in order to avoid the blocking effect of the CMAC neural network.

From the above discussion, it is very attractive to combine the CMAC neural network architecture and learning methods into the design of fuzzy logic controller because this combination overcomes the problem of long learning time of the existing MLP based FLC due to the inherent local computation of the CMAC. Some examples of such researches can be founded in the following. Jou [8] proposed an adaptive fuzzy controller, called FCMAC, that was built in the framework of the CMAC network. He argued that the FCMAC could be used as a model-free function estimator. Unfortunately, he presented only the conceptual work and did not show any experimental results. Ker et al. [9] proposed to use the fuzzy CMAC model for color reproduction. In their model, recursive B-spline receptive field functions were replaced by fuzzy sets with bell-shaped membership function and the output weights were fuzzy variables. It showed a very effective color reproduction, a fast learning, a simple computation, and a high stability on model parameters.

This research proposes a new CMAC-based FLC, called CBFLC, that is a fuzzy logic controller whose architectural structure is based on the CMAC model. When combining the CMAC network model into FLC, one problem is the lower approximation ability and control performance that is caused by the improper selection of model parameters are due to the CMAC's local learning. Also, most existing CMAC-based FLCs adjust only output weights that are corresponding to the singleton values of defuzzifier module. But, many researchers have been addressed to adjust the structural parameters of both input and output membership functions in order to improve the approximation or control performance of the FLC [3,10]. This fact motivates that unlike the existing CMAC-based FLC model, the proposed CBFLC should adjust the input and output membership functions together and each membership function should be characterized by two model parameters (the center and width). This co-adaptation of input and output membership functions and the extension of model parameters will improve the approximation ability and the control performance. Further, the model

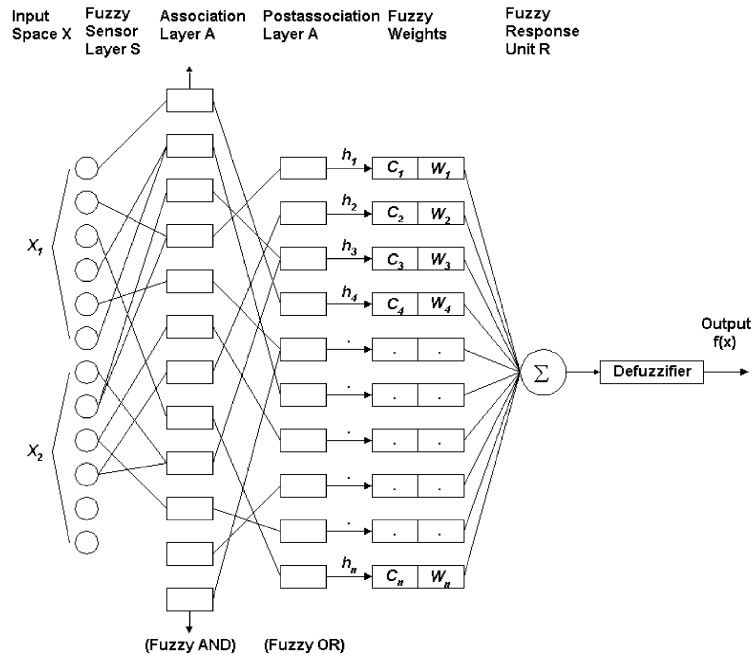


Fig. 1. Structure of the CMAC-based FLC.

parameters are trained by the selective backpropagation learning where only the activated units participate in the error backpropagation. This will make the learning procedure faster than the global weight updating of backpropagation learning.

This paper is organized as follows. Section 2 explains the architecture and operation of the proposed CBFLC and presents a learning scheme of determining the optimal structural parameters. Section 3 applies the proposed CBFLC to the truck backer-upper control problem and compares various performances with other FLCs. Finally, a conclusion is drawn.

2. CMAC-based fuzzy logic controller

2.1. Architecture and operation

When the input $x = (x_1, x_2, \dots, x_n)$ in X is the n -dimensional continuous-valued vector and the $f(x)$ is the scalar output corresponding to the input vector, the proposed CBFLC can approximate any arbitrary nonlinear input–output function by the following three consequent mappings as in [5].

$$\begin{aligned}
 \Theta &: X \rightarrow S, \\
 \Phi &: S \rightarrow A, \\
 \Psi &: A \rightarrow P,
 \end{aligned} \tag{1}$$

where S, A and P represent three layers with different number of units. Fig. 1 shows a structure of the proposed CMAC-based FLC.

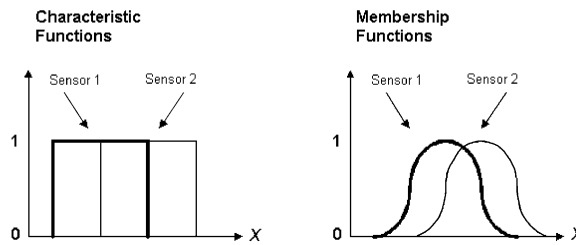


Fig. 2. Response characteristics of two different sensors.

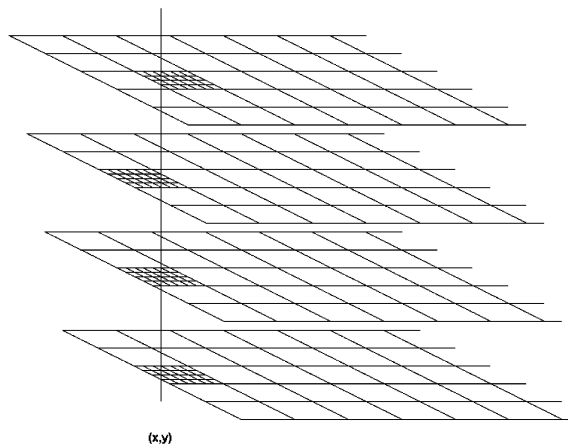


Fig. 3. Fuzzy sensor layer S with multiple layers.

The first mapping ($\Theta : X \rightarrow S$) performs a fuzzification that transforms a received crisp input into a given number of fuzzified outputs. Each input variable x_i of the input vector x is associated with a set of sensors S_{ij} ($j = 1, 2, \dots, |x_i|$), where $|x_i|$ means the number of fuzzy sensors that the input variable x_i has. The input sensor of the regular CMAC neural network gives a binary output depending on whether the value of input variable x_i is located within the receptive field $[x_{ij} - \delta, x_{ij} + \delta]$ of the sensor S_{ij} . However, each sensor S_{ij} of the proposed CMAC-based FLC has a bell-shaped response that includes the fuzziness into the activity of the sensor. We call such a sensor the *fuzzy sensor*. So, each fuzzy sensor s_{ij} is characterized by a membership function $\mu_{S_{ij}}(x_i)$ that is characterized by two model parameters (x_{ij}, s_{ij}) , where x_{ij} and s_{ij} are the center and width of the membership function, respectively. When only two membership functions are allowed to be overlapped, each input produces two fuzzified outputs maximally. Fig. 2 compares the different responses of the regular and fuzzy sensors.

Often, multiple sensor layers have been used to reduce the blocking effect and to improve the approximation ability of the CMAC as shown in Fig. 3, where each layer has identical organization but are offsets relative to the others. When the fuzzy sensor layer S consists of l layers and the maximum overlapping among the sensors in each individual layer is m , an n -dimensional input $x = (x_1, x_2, \dots, x_n)$ produces m^l fuzzy outputs. We note that four fuzzy sensors responded to the input (x_1, x_2) when $l = 4$ and $m = 1$ because four layers are located with different offsets.

The second mapping ($\Phi : S \rightarrow A$) performs a fuzzy AND operation that selects the minimum value from the incoming n fuzzy outputs, where each fuzzy output comes from each input variable. When the input space is n

dimensional, the number of fuzzy sensors of the input variable x_i is $|x_i|$, each sensor layer consists of l layers, and the degree of overlapping among fuzzy sensors is m , the total number of fuzzy AND units is $\prod_{i=1}^n |x_i| m^n l$. When an input (x_1, x_2, \dots, x_n) is applied, $m^n l$ fuzzy sensors are activated maximally. Among the fuzzy AND units in the layer A , only $m^n l$ fuzzy AND units that receive n activated fuzzy sensor outputs are activated. The activated fuzzy AND units in the layer A perform the MIN operation and store the index information of fuzzy sensor that has the smallest value among the incoming values, where the index information will be later in the error backpropagation.

The third mapping ($\Psi : A \rightarrow P$) performs a fuzzy OR operation that selects the maximum value among the incoming fuzzy AND outputs, where each fuzzy AND output is randomly connected to some unit in the layer P . Generally, the number of fuzzy OR units in the layer P is much smaller than the number of fuzzy AND units in the layer A because the available data in the real world is much smaller than all possible data ($|P| \ll |A|$). Also, it is common that the number of activated fuzzy OR units in the layer P is smaller than the number of activated fuzzy AND units in the layer A ($m^n l$) due to the possibility of collision by the random mapping from the layer A to the layer P . Each unit in the layer P performs the MAX operation that selects the one having the greatest value among the incoming fuzzy AND outputs from the layer A . This selection can be acted as an avoiding means of the collision problem that many units of different virtual addresses in the layer A are mapping to the same unit of a physical address in the layer P . We need to store an index information of the unit in the layer A that passes the largest value to each unit in the layer P in order to use it later in the error backpropagation.

The output $\mu_{p_i}(\mathbf{x})$ of a fuzzy OR unit in the layer P forms a fuzzy relation $\mu_{p_i}(\mathbf{x}) \rightarrow \mu_{w_i}(y)$ ($i = 1, 2, \dots, |P|$) with a corresponding physical memory that stores the weight constant $\mu_{w_i}(y)$ of the bell-shaped membership function, where the membership function is characterized by the center and width (y_i, s_i) . When a fuzzy OR unit P^* is activated by a given input, the output $\mu_R(y)$ of fuzzy response unit R is represented by the following equation as

$$\mu_R(y) = \bigvee_{p_i \in P^*} \mu_{p_i}(\mathbf{x}) \wedge \mu_{w_i}(y), \tag{2}$$

where \wedge is a fuzzy AND operator that is often performed by the MIN operation, $\mu_{p_i}(\mathbf{x}) \wedge \mu_{w_i}(y)$ represents the mutual compatibility between the precondition and consequence of a fuzzy rule, and $\bigvee_{p_i \in P^*}$ is a fuzzy OR operator that is often performed by the MAX operation. So, the above equation implies that the proposed CBFLC takes the greatest one among the selected minimum values.

Since many fuzzy responses $\mu_R(y)$ are distributed over the output space, we need to take a defuzzification operation in order to obtain a single crisp output. Among many defuzzification methods, we take the center of gravity (COG) defuzzification method as

$$y_c = \frac{\sum_{i=1}^{|P^*|} y_i \mu_R(y_i)}{\sum_{i=1}^{|P^*|} \mu_R(y_i)}. \tag{3}$$

However, the above COG defuzzifier uses the clipped membership values at the singletons in computing the defuzzified crisp value. This fails to reflect our instinct that the membership function having the larger area should be outweighed. To overcome this failing, we propose a new defuzzification method that reflects the area of the membership function as [10]

$$y_c = \frac{\sum_{i=1}^{|P^*|} y_i \mu_R(y_i) s_i}{\sum_{i=1}^{|P^*|} \mu_R(y_i) s_i}. \tag{4}$$

2.2. Learning

To learn the proposed CBFLC, we have modified the conventional backpropagation learning [12] as follows. First, the output error is used to adjust the structural parameters (centers and widths) of the input and output membership functions while it is used to adjust the weights of preceding units in the original backpropagation learning. Next, the modified backpropagation learning adjusts the structural parameters of only some activated units while the original backpropagation learning adjusts all weights in the network. Detailed explanation of the backpropagation learning of the proposed CBFLC can be given by the following.

Assume that a training example set $T = \{e^1, e^2, \dots, e^N\}$ consisting of N samples has been known in advance. An example e^t ($t = 1, 2, \dots, N$) consists of $(x_1^t, x_2^t, \dots, x_n^t; y^t)$, where x_i^t is the i th-dimensional input value of the t th example and y^t is an output value of the t th example. The proposed CBFLC produces a defuzzified output y_c^t for a given input $(x_1^t, x_2^t, \dots, x_n^t)$. Then, the system error E of the CBFLC over the training example set T is computed by

$$E = \frac{1}{2} \sum_{t=1}^N (y^t - y_c^t)^2 = \frac{1}{2} \sum_{t=1}^N \left(y^t - \frac{\sum_{i=1}^n y_i \mu_R(y_i) s_i}{\sum_{j=1}^n \mu_R(y_j) s_j} \right)^2, \tag{5}$$

where n , N , and $\mu_R(y_i)$ are the number of output fuzzy terms, the number of training examples, and the clipped membership value of the i th output fuzzy term, respectively.

The center and width (y_i, s_i) of the output membership function of the activated units in the layer P are adjusted by the generalized delta rule [11] and a simple chain-rule of differentiations as

$$\begin{aligned} y_i(t+1) &= y_i(t) - \eta_i^y \sum_{t=1}^N (y^t - y_c^t) \frac{\mu_R(y_i) s_i}{\sum_{j=1}^{|P^*|} \mu_R(y_j) s_j}, \\ s_i(t+1) &= s_i(t) - \eta_i^s \sum_{t=1}^N (y^t - y_c^t) \frac{\mu_R(y_i) (\sum_{j=1}^{|P^*|} \mu_R(y_j) s_j (y_i - y_j))}{(\sum_{j=1}^{|P^*|} \mu_R(y_j) s_j)^2}, \end{aligned} \tag{6}$$

where η_i^y and η_i^s are the learning rates of the center and width of the i th output fuzzy term.

The system error E in the output layer is backpropagated to the activated units in the layer P . The backpropagated error E_P^i of the i th activated unit in the layer P is computed as

$$E_P^i = -\frac{\partial E}{\partial \mu_R(y_i)} = -\sum_{t=1}^N \frac{\partial E^t}{\partial y^t} \frac{\partial y^t}{\partial \mu_R(y_i)} = -\sum_{t=1}^N (y_t - y_c^t) \left\{ \frac{y_i s_i (\sum_{j=1}^{|P^*|} \mu_R(y_j) s_j) - (\sum_{j=1}^{|P^*|} y_j \mu_R(y_j) s_j) s_i}{(\sum_{j=1}^{|P^*|} \mu_R(y_j) s_j)^2} \right\}. \tag{7}$$

The backpropagated error E_P^i to the layer P will be directly passed to the fuzzy OR units in the layer A that activate the fuzzy AND units in the layer P . So, the error E_A^i of the activated fuzzy OR units in the layer A is exactly same as the error E_P^i . The backpropagated error E_A^i to the layer A will be passed to the fuzzy sensors in the layer S that has the smallest value among the fuzzy sensors incoming to the activated fuzzy AND unit.

Assume that the j th fuzzy sensor of the i th input variable has the bell-shaped membership function $f_{ij} = \exp(-(1/2)(x_i - x_{ij})^2/s_{ij}^2)$. Then, two structural parameters of the fuzzy sensors are adjusted by the generalized delta rule and a simple chain-rule of differentiations as

$$x_{ij}(t+1) = x_{ij}(t) + \eta_{ij}^x \sum_k e_k f_{ij}(x_k) \frac{(x_k - x_{ij})}{s_{ij}^2},$$

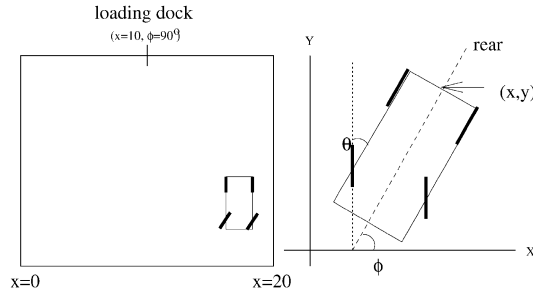


Fig. 4. A model truck and a loading dock in the truck control.

$$s_{ij}(t + 1) = s_{ij}(t) + \eta_{ij}^s \sum_k e_k f_{ij}(x_k) \frac{(x_k - x_{ij})^2}{s_{ij}^3}, \tag{8}$$

where

$$e_k = \begin{cases} E_A^k & \text{when the } j\text{th sensor output is the smallest among the inputs of the } k\text{th fuzzy AND unit,} \\ 0 & \text{otherwise.} \end{cases}$$

The above learning procedure will be continued until the system error becomes sufficiently small (i.e., $E < \varepsilon$) or the iteration is sufficiently large (i.e., $t > T$), where ε and T are the threshold values of system error and iteration, respectively.

3. Simulation results and discussion

The proposed CBFLC is applied to the truck backer-upper control in order to verify how well it is operating. The goal of the truck backer-upper control problem is to back a truck to a loading dock as quickly and precisely as possible. This control problem is a typical non-linear control problem that cannot be solved by the conventional control techniques. Fig. 4 shows a model truck and a loading dock used in the truck backer-upper control problem. The position of the truck is precisely determined by (x, y, ϕ) where ϕ is an angle between the truck's onward direction and the x axis, and the tracking control of the truck is done by the θ where θ is an angle between the truck's onward direction and the axis of wheel. The approximate control dynamics of the truck backer-upper control problem is given as (See [13] for details.)

$$\begin{aligned} x(t + 1) &= x(t) + \cos[\phi(t) + \theta(t)] + \sin[\theta(t)] \sin[\phi(t)], \\ y(t + 1) &= y(t) + \sin[\phi(t) + \theta(t)] - \sin[\theta(t)] \cos[\phi(t)], \\ \phi(t + 1) &= \phi(t) - \sin^{-1} \left[2 \sin \left(\frac{\theta(t)}{b} \right) \right], \end{aligned} \tag{9}$$

where b is the length of truck and $b = 4$ is taken in this work. When the distance between the truck and the loading dock is sufficiently great, it has only to back the truck straightforwardly once the truck comes close to near $x = 10$ and $\phi = 90^\circ$. Thus, the variable y can be excluded from the fuzzy input variables (x, y, ϕ) for simplicity. So, the design problem of FLC for the truck backer-upper control problem is thought as to back the truck at a certain position (x_o, ϕ_o) in the interval of $\{0 \leq x \leq 20, -90^\circ \leq \phi \leq 270^\circ\}$ to the loading dock at $x = 10, \phi = 90^\circ$ as fast and precisely as possible.

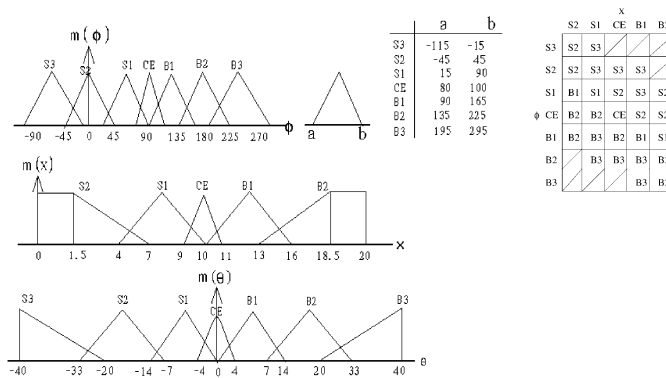


Fig. 5. Membership functions (left) and fuzzy rule base (right) used in Wang’s work.

Table 1
Structural and learning parameters for the proposed CBFLC

	No. of layers = 1	No. of layers = 4
Fuzzy variables	$(x, \phi; \theta)$	$(x, \phi; \theta)$
No. of MFs	(5, 7; 1024)	(20, 28; 4096)
No. of activated MFs	(2, 2; 4)	(8, 8; 16)
Size of virtual memory	65 536	65 536
Size of physical memory	1024	4096
No. of training examples	236	236
Learning rates	(0.5, 0.5; 0.5)	(0.5, 0.5; 0.5)
No. of iterations	200	200

Fig. 5 shows the membership functions of the fuzzy input and output variables and the fuzzy rule base used in the Wang and Mendel’s work. We will use the same naming convention of linguistic variables as Wang’s work in a way that (S2, S1, CE, B1, B2) for 5 partitions and (S3, S2, S1, CE, B1, B2, B3) for 7 partitions. We used the same training example set as in the Wang and Mendel’s work. This training example set consists of 238 input–output data pairs that consist of the smooth trajectories from 14 initial positions to the loading dock. (See [13, Tables 1–14] for details.)

The structural and learning parameters of the CBFLC used in the simulation is represented in Table 1. To make a fair comparison with the Wang’s work [14], the input variables x and ϕ are partitioned into 5 or 7 fuzzy terms. The conventional CMAC neural network requires the infinite number of virtual memories theoretically. When the interval of each input x_i is $[0, |x_i|]$ and the resolution of input is δ , the input x_i is quantized into $N_\delta = |x_i|/\delta$ intervals. Therefore, it is sufficient to take $(N_\delta)^n$ cells as the size of virtual memory in the case of n dimensional inputs. In this work, we take 256×256 cells as the size of the virtual memory because the dimension of inputs is two and each input is divided into 256 intervals.

We compare the learning time of the proposed CBFLC with that of the MLP-based FLC (MBFLC) as follows. First, compare the number of iterations between them. Let the number of learning iterations of the MBFLC and CBFLC be N_{MBFLC} and N_{CBFLC} , respectively. As mentioned earlier, CBFLC pursues the local generalization of the input spaces while MBFLC pursues the global generalization of the input spaces. So, we found that $N_{CBFLC} \ll N_{MBFLC}$ due to the easiness of learning of the CBFLC. Fig. 7 shows the learning curves of the proposed CBFLC that is applied to the truck backer-upper control problem. From this figure, we note that the learning error is convergent approximately within 200 iterations. But, the MBFLC proposed by

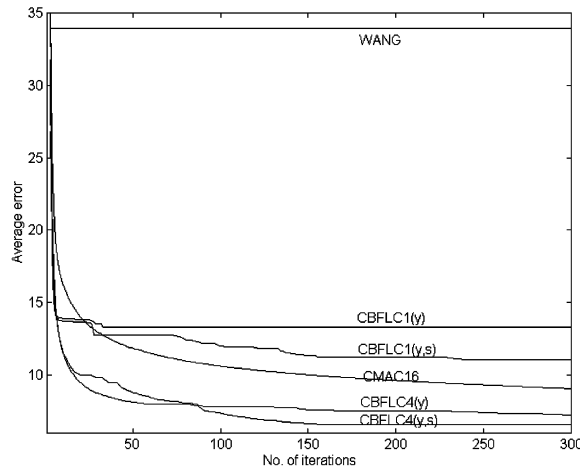


Fig. 6. Comparison of approximation errors among many different FLCs.

Lin [3] showed that the learning error was convergent after 2000 iterations when it was applied to the same problem. From these two simulation results, we note that $N_{CBFLC} \approx 10^{-1} N_{MBFLC}$. Next, compare the amount of computation per each learning iteration between them. Assume that the number of units in each layer of two FLCs are equal each other. Operational process in each unit of two FLCs are similar in the forward and backward computation, but the number of units that participated in the operation is quite different. While all units in the layer are participated into the forward and backward computation in the case of MBFLC, only some activated units are involved in the forward and backward computation in the case of CBFLC. Therefore, we note that the amount of computation of the CBFLC is much smaller than that of the MBFLC. From these two facts we can conclude that the proposed CBFLC can be learned more rapidly than the MBFLC.

Next, we compare the approximation ability among many FLCs in terms of the average system error $E = \sum_{t=1}^{236} \frac{1}{2} (y^t - y_c^t)^2$. In this work, we used the same training sample set of 236 samples that was taken in the Wang’s work [13]. In our work, six different FLCs are considered as follows.

- (1) Wang’s non-learned FLC (WANG),
- (2) CMAC neural network with 16 input layers (CMAC16),
- (3) CBFLC with the conventional COG defuzzifier and single fuzzy sensor layer (CBFLC1(y)),
- (4) CBFLC with the conventional COG defuzzifier and four fuzzy sensor layers (CBFLC4(y)),
- (5) CBFLC with the proposed COG defuzzifier and single fuzzy sensor layer (CBFLC1(y,s)),
- (6) CBFLC with the proposed COG defuzzifier and four fuzzy sensor layers (CBFLC4(y,s)).

In order to avoid the parameter tweak, we performed five runs of simulations of each FLC. Fig. 6 shows that the average approximation errors at the 300 iteration are 6.60, 7.22, 9.07, 11.08, 13.31, and 34.0 in the order of CBFLC4(y,s), CBFLC4(y), CMAC16, CBFLC1(y,s), CBFLC1(y), and WANG, respectively. From these results, we note that we can get better approximation ability in the case of learned FLC than the non-learned FLC, in the case of having many fuzzy sensors under the same CBFLC, and in the case of adjusting the center and width of the COG defuzzifier simultaneously.

Next, we compare the control performances of six different FLCs mentioned above. The control performances were evaluated in terms of the average tracing distance $d(T) = \frac{1}{100} \sum_{t=1}^{100} \sum_{s=1}^{S_t} d_{ts}$ from the randomly chosen 100 starting positions in the intervals ($\{0 \leq x \leq 20, 0 \leq y \leq 50, -270^\circ \leq \phi \leq 90^\circ\}$) to the goal, where S_t is the number of steps of the t th starting sample reaching to the goal and d_{ts} is the distance of the t th sample during the s th step that is computed by $d_{ts} = \sqrt{(x_{t,s+1} - x_{t,s})^2 + (y_{t,s+1} - y_{t,s})^2}$. Fig. 7 shows that the average tracing distances are 17.73, 20.72, 21.91, 22.53, 24.09, 27.89, and 39.92 in the order of CMAC16, CBFLC4(y,s),

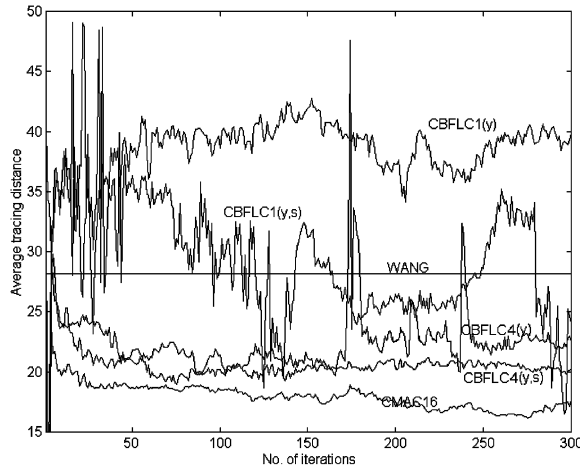


Fig. 7. Comparison of average tracing distances among different FLCs.

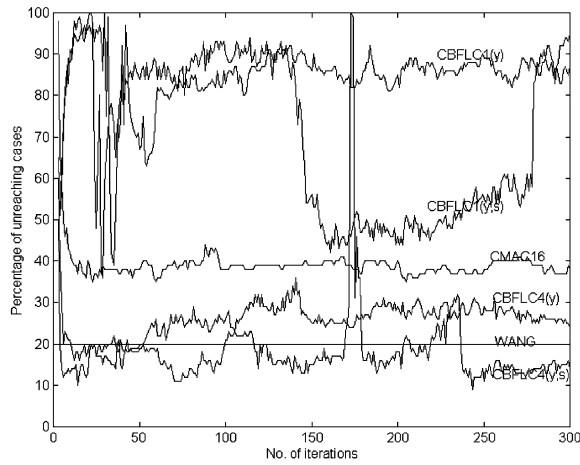


Fig. 8. Comparison of non-reaching ratios to the goal among different FLCs.

CBFLC4(y), CBFLC1(y,s), WANG, and CBFLC1(y), respectively. From these results, we note that (1) the number of the input fuzzy sensor layers makes a great effect on the control performance in that the CMAC16 having the largest number of input fuzzy sensor layers produces the shortest average tracing distance, (2) the CBFLC with the simultaneous adjustment of the MF’s center and width produces better control performance than the CBFLC with the adjustment of only the MF’s center and (3) the WANG’s FLC having the non-learned MFs produces a worse control performance because their fuzzy rule base and MFs were determined to produce their control trajectories as smooth as possible.

Next, we compare the reaching performance to the goal of six difference FLCs. For doing this, we define the non-reaching ratio to the goal $r = 1 - N_{\text{reach}}/N_{\text{test}}$ from the randomly chosen 100 starting positions in the intervals ($\{0 \leq x \leq 20, 0 \leq y \leq 50, -270^\circ \leq \phi \leq 90^\circ\}$) to the goal, where N_{reach} and N_{test} are the number of reaching samples to the goal and the number of test samples (= 100), respectively. Fig. 8 shows that the non-reaching ratios are 14 in the order of CBFLC4(y,s), WANG, CBFLC4(y), CMAC16, CBFLC1(y,s),

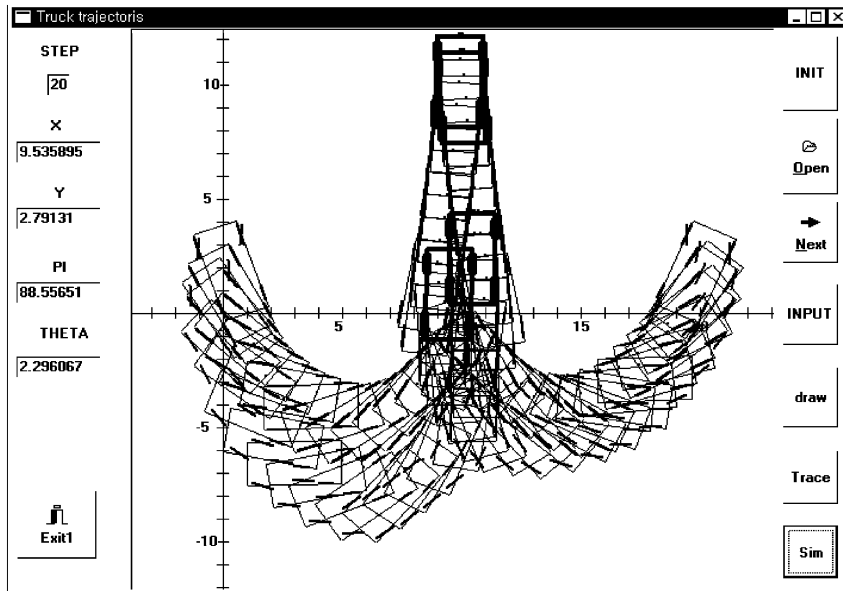


Fig. 9. Comparison of trajectories among different FLCs.

and $CBFLC1(y)$, respectively. From these results, we note that (1) the proposed COG defuzzifier with the simultaneous adjustment of the MF's center and width has the superior generalization ability in that the $CBFLC4(y,s)$ produces the lowest non-reaching ratio, (2) the CBFLC with the simultaneous adjustment of the MF's center and width produces better reaching performance to the goal than the CBFLC with the adjustment of only the MF's center and (3) the WANG's FLC having the non-learned MFs produce a relatively good control performance because their fuzzy rule base and MFs were determined to produce their control trajectories as smooth as possible.

Fig. 9 compares the tracing trajectories of the model car starting from (a) $(1.0, 0.0, -70^\circ)$ and (b) $(19.0, 0.0, -245.0^\circ)$ of both the non-learned Wang's FLC. The trajectories that advance to the lower position and reach to the more front position are for the $CBFLC4(y,s)$ and the trajectories that advance to the upper position and reach to the more back position are for the Wang's non-learned FLC. We found that the number of steps of the $CBFLC4(y,s)$ are 20 (for starting at the left side) and 19 (for starting at the right side) and 2) the numbers of steps of the Wang's non-learned FLC are 27 (for starting at the left side) and 25 (for starting at the right side). These differences come from the fact that the proposed $CBFLC4(y,s)$ has the new COG defuzzifier with the simultaneous adjustment of the MF's center and width and the defuzzified output becomes large due to the additional weight by the width term. So, this reinforced defuzzified output turns the wheel's angle θ much as θ becomes larger.

4. Conclusion

This paper proposed a new type of CMAC-based FLC that combines the functionality of FLC into the architecture of CMAC neural network. It can improve the long learning time of the conventional MLP-based FLC and the inaccurate approximation ability of the CMAC neural network. The proposed CBFLC can learn faster than the conventional MLP-based FLC because the CBFLC pursues local generalization that is easy to

learn and only some activated units are participated in learning while the MLP-based FLC pursues the global generalization that is hard to learn and all units in the MLP are participated in learning.

The conventional COG defuzzifier in the CMAC neural network uses only singleton values of output variable and they are learned by the well-known LMS adaptation rule under the assumption that the units are operating linearly. So, the conventional CMAC-based FLC produced the lower approximation ability. We overcome this disadvantage by proposing a new type of COG defuzzification method that uses both MF's center and width in computing a defuzzified crisp value and by adjusting the structural parameters of the input and output variables together.

The proposed CBFLC was applied to the truck backer-upper control problem in order to test its performances. We compared various aspects such as the approximation ability, the control performance, and the reaching performance to the goal in terms of approximation error, average tracing distance and the reaching ratio to the goal, respectively, among many different FLCs such as Wang's non-learned FLC, CMAC neural network, MLP-based FLC, and the proposed CBFLC. Simulation results verified that the proposed CBFLC produces superior performance in all aspects such as the approximation ability, and control performance, and the reaching performance. Presently, we are performing the implementation work of the proposed CBFLC that will be applied to control the mobile Khepera robot on the reconfigurable FPGA system [2].

References

- [1] J.S. Albus, A new approach to manipulator control: the cerebellar model articulation controller (CMAC), *J. Dynamic Systems Measurement Control* 97 (3) (1975) 220–227.
- [2] S. Casselman, M. Thornburg, J. Schewel, Hardware Object Programming on the EVC1: a reconfigurable computer, FPGAs for Rapid Board Development and Reconfigurable Computing (*Photonics East 95*), (1995) 191–199.
- [3] C. Lin, C.S. Lee, Neural-network-based fuzzy logic control and decision system, *IEEE Trans. Comput.* 40 (12) (1991) 1320–1336.
- [4] E.H. Mandani, Application of fuzzy algorithms for control of simple dynamic plant, *IEEE Proc. Control Sci.* 121 (12) (1974) 1585–1588.
- [5] W.T. Miller, F.H. Glanz, L.G. Kraft, CMAC: an associative neural network alternative to backpropagation, *Proc. IEEE* 78 (10) (1990) 1561–1567.
- [6] J. Moody, Fast learning in multi-resolution hierarchies, in: D.S. Touretzky (Ed.), *Advances in Neural Information Processing Systems*, Morgan Kaufmann, Los Altos, CA, 1989, pp. 29–39.
- [7] J.R. Jang, ANFIS: adaptive-network-based fuzzy inference system, *IEEE Trans. Systems Man Cybernet.* 23 (2) (1993) 665–685.
- [8] C.-C. Jou, A fuzzy cerebellar model articulation controller, *IEEE Internat. Conf. on Fuzzy Systems*, 1992, pp. 1179–1186.
- [9] J.S. Ker, C.C. Hsu, Y.H. Kuo, B.D. Liu, A fuzzy CMAC model for color reproduction, *Fuzzy Sets and Systems* 91 (1) (1997) 53–68.
- [10] D. Kim, Improving the fuzzy system performance by fuzzy system ensemble, *Fuzzy Sets and Systems* 98 (1) (1998) 43–56.
- [11] H.S. Rhee, K.W. Oh, A design and analysis of objective function-based unsupervised neural networks for fuzzy clustering, *Neural Process. Lett.* 4 (2) (1996) 83–95.
- [12] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, *Parallel Distributed Processing: Explorations in Microstructures of Cognition*, MIT Press, Cambridge, MA, 1988.
- [13] L.X. Wang, J.M. Mendel, Generating fuzzy rules from numerical data, with applications, USC-SIPI Report, 1991, p. 169.
- [14] L.X. Wang, J.M. Mendel, Generating fuzzy rules by learning from examples, *IEEE Trans. Systems Man Cybernet.* 22 (6) (1992) 1414–1427.
- [15] Y. Wong, A. Sideris, Learning convergence in the cerebellar model articulation controller, *IEEE Trans. Neural Networks* 3 (1) (1992) 111–121.