2009 American Control Conference
Hyatt Regency Riverfront, St. Louis, MO, USA
June 10-12, 2009

Genetic Based Fuzzy Logic Controller For a Wall-Following Mobile Robot

Sameh F. Desouky and Howard M. Schwartz

Abstract— **This paper addresses the problem of tuning fuzzy logic controllers. In this paper we presents a new technique called a genetic based fuzzy logic controller (GBFLC). The proposed technique is used to iteratively tune the set of fuzzy logic controller parameters such as membership functions and scaling factors. The proposed technique is also used to reduce the number of fuzzy rules. Computer simulations are performed on a wall-following mobile robot and the results show the usefulness of the proposed technique.**

I. INTRODUCTION

Fuzzy logic controllers (FLCs) are currently being used to a great extend in engineering applications [1], [2] especially for plants that are complex and ill-defined [3], [4] and plants with high uncertainty in the knowledge about its environment such as autonomous mobile robotic systems [5]. However, FLC has a drawback of finding its knowledge base which is based on a tedious and unreliable trial and error process [6]. One way of dealing with this problem is to tune the set of FLC parameters through the use of genetic algorithms (GAs).

GAs have been used to overcome the difficulty and complexity in the tuning of the FLC parameters such as membership functions (MFs), scaling factors and control rule configurations [7]. Unlike many classical optimization techniques, GAs do not rely on computing local derivatives to guide the search process; all they need is an objective function [8].

In this paper, a two stage tuning process is proposed to tune the parameters of FLC such as MFs and scaling factors, and to reduce the number of the fuzzy rules. In stage 1, GAs tune the FLC parameters using input/output data pairs obtained from a PD controller which is used as the initial expert. In stage 2, the PD controller and the FLC tuned in stage 1 operate in parallel and then we get new input/output data pairs which are used by GAs to tune the parameters of the FLC again. Stage 2 is repeated until the GBFLC has achieved the desired performance.

This paper is organized as follows: in Section II the dynamics of a mobile robot that follows a wall are obtained, also some basic terminologies for the FLC and GAs are reviewed. Section III describes our proposed GBFLC technique. Computer simulations are presented in Section IV and the results are discussed in Section V. Finally, conclusions are pointed out in section VI.

Sameh F. Desouky is with Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada sameh@sce.carleton.ca

Howard M. Schwartz is with the Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada schwartz@sce.carleton.ca

II. PRELIMINARIES

A. Mobile robot dynamics

We want the mobile robot to follow a wall at a desired distance. The minimum distance to the wall is the perpendicular line from the robot to the wall as shown in Fig. 1. Equations of motion for a mobile robot are [9], [10]

$$\dot{x}_m = V_m \cos\theta_m \tag{1}$$

$$\dot{y}_m = V_m \sin\theta_m \tag{2}$$

$$\dot{\theta}_m = \frac{V_m}{l}\tan u \tag{3}$$

where x_m, y_m are the cartesian distances between the robot and the wall, θ_m is the orientation of the robot, V_m is the velocity, l is the turning radius, and u is the steering angle which is the control action.

B. PD Controller Design

Fig. 2 shows a block diagram of a PD controller system. The input to the PD controller is the error in distance from the robot to the wall, and is calculated as

$$e = D_d - D_m \tag{4}$$

where D_d and D_m are the desired and actual distances between the robot and the wall, respectively. The output of the PD controller is the action, u, which is defined as

$$u = k_p\, e + k_d\, \dot{e} \tag{5}$$

where k_p and k_d are the proportional and the differential gain coefficients, respectively.

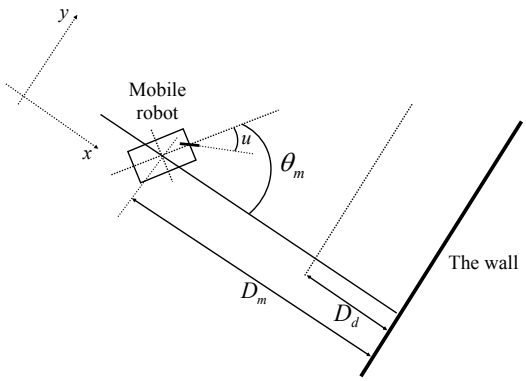

Fig. 1. Dynamics of a wall-following mobile robot

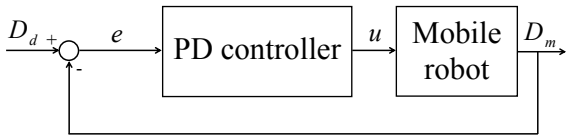

Fig. 2. Block diagram of a PD controller system

C. FLC

A block diagram of a FLC is shown in Fig. 3. It has two inputs, the error, e, and the derivative of the error, $\dot{e}$, and its output is the control action, u. The FLC is used to control the distance of a mobile robot to a wall.

We choose 5 trapezoidal MFs for the inputs, e and $\dot{e}$, and for the output, u, with the same linguistic values HN, LN, Z, LP and HP where **HN** means **H**igh **N**egative, **LN** means **L**ow **N**egative, **Z** means **Z**ero, **LP** means **L**ow **P**ositive, and **HP** means **H**igh **P**ositive. A trapezoidal MF which is shown in Fig. 4 is described by

$$\mu(x) = \begin{cases} 0 & : \quad x < a \\ \dfrac{x-a}{b-a} & : \quad a \le x < b \\ 1 & : \quad b \le x < c \\ \dfrac{x-c}{b-c} & : \quad c \le x \le d \\ 0 & : \quad x > d \end{cases} \tag{6}$$

where a, b, c, and d are its coefficients. The fuzzy inference system used in the FLC is a Mamdani type which has rules of the form

$$R_l : IF\ e\ is\ A_1^l\ AND\ \dot{e}\ is\ A_2^l\ THEN\ u_l\ is\ C_l,\ w_l = q_l \tag{7}$$

where A_i^l is the fuzzy set of the i^{th} input in rule l, C_l is the fuzzy set of the output u_l in rule l, $l = 1,2,\ldots,25$, w_l is the weight of the l^{th} rule and $q_l \in [0,1]$. Table I represents

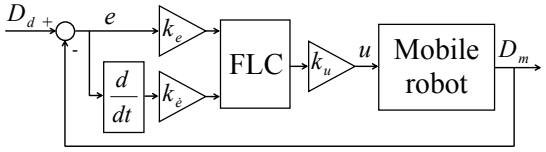

Fig. 3. Block diagram of a FLC system

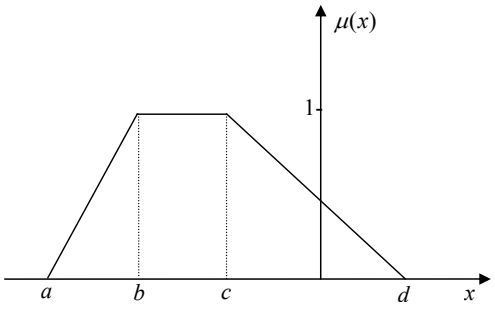

Fig. 4. A trapezoidal membership function

TABLE I
FUZZY DECISION TABLE

e \ $\dot{e}$	HN	LN	Z	LP	HP
HN	HN	HN	HN	LN	Z
LN	HN	HN	LN	Z	LP
Z	HN	LN	Z	LP	HP
LP	LN	Z	LP	HP	HP
HP	Z	LP	HP	HP	HP

the fuzzy decision table for the inputs e and $\dot{e}$. The crisp output can be calculated using the center of area (COA) deffuzification method as follows

$$u = k_u \frac{\sum_{l=1}^{m} \mu^{C_l}(u_l).u_l}{\sum_{l=1}^{m} \mu^{C_l}(u_l)} \tag{8}$$

where k_u is the scaling factor for the output, and m is the number of fuzzy rules in the system.

Due to normalization of the input and the output variables, we need to use the scaling factors. We denote the scaling factors of the inputs as k_e and $k_{\dot{e}}$, and of the output as k_u.

Since the tuning of scaling factors is a trial-and-error process, it takes much time to reach its best values. These values may be good but may not be the optimum. In this paper, we tune the scaling factors of the inputs and the output using the proposed GBFLC.

D. GAs

GAs, which were proposed by Holland in 1973 [11], are search and optimization techniques that are based on a formalization of natural genetics [12].

GAs search a multidimensional parameter space to find an optimal solution. A given set of parameters is referred to as a chromosome. The parameters can be either decimal or binary numbers. The GA is initialized with a number of randomly selected parameter vectors or chromosomes. This set of chromosomes is the initial population. Each chromosome is tested and evaluated based on a fitness function; in control engineering we would refer to this as a cost function. The chromosomes are sorted based on the lowest cost function or the ranking of the fitness functions. One then selects a number of the best, according to the fitness function, chromosomes to be parents of the next generation of chromosomes. A new set of chromosomes is selected based on reproduction.

In the reproduction process, we generate new chromosomes, which are called children. We use two GA operations. The first operation is a crossover in which we choose a pair of parents and select a random point in all of their chromosomes and make a cross replacement from one parent to another. The second operation is a mutation in which a parent is selected and we change one or more of its parameters to get

a new child. Now, we have a new population to test again with the fitness function. The genetic process is repeated until the last iteration is reached.

III. The proposed GBFLC technique

Our proposed GBFLC is shown in Fig. 5. Tuning the parameters of the FLC passes through two stages. In stage 1, we run the PD controller as shown in the upper figure to obtain the initial input/output data pairs. Then we use GAs to tune the parameters of the FLC using these input/output data pairs. In Stage 2, the PD controller and the FLC tuned in stage 1 operate in parallel as shown in the lower figure. We then get new input/output data pairs which are used by GAs to tune the parameters of the FLC again. Stage 2 is repeated until the FLC has achieved the desired accuracy. Algorithm 1 describes the tuning process in stage 1 and Algorithm 2 describes the tuning process in stage 2. In [13], a similar technique is used but the training data is used to tune the weights of a neural network controller.

Algorithm 1 : Tuning in stage 1

1) Run the PD controller and obtain L input/output data pairs.
2) Initialize population with size P for the MFs parameters, the scaling factors, and the fuzzy rules' weights chromosomes.
3) Repeat for each iteration i
   a) repeat for each population p
      i) Construct a FLC from the p^{th} chromosome of the MFs parameters, the scaling factors, and the fuzzy rules' weights.
      ii) Repeat for each input/output data pair l
      - Evaluate the output, u^l_{flc}, of the constructed FLC using the inputs obtained in step 1.

      iii) Calculate the p^{th} fitness function. We use the mean square error function (MSE) as the fitness function which is defined as

$$MSE = \frac{1}{2L}\sum_{l=1}^{L}(u^l_d - u^l_{flc})^2 \qquad (9)$$

   b) Sort the entire chromosomes (for the MFs parameters, the scaling factors, and the fuzzy rules' weights) of the population according to their fitness values.
   c) Select a portion of the sorted population as the new parents.
   d) Create a new generation for the remaining portion of population using crossover and mutation.

IV. Computer Simulation

We choose the velocity of the robot to be constant, $V_m = 50$ cm/s. The robot starts moving with an initial orientation, $\theta_m = 0$ rad. The turning radius, $l = 10$ cm. The desired distance between the robot and the wall is $D_d = 100$ cm. We choose the proportional and the differential gains of the PD controller k_p and k_d to be 1.2 and 1.5, respectively.

Algorithm 2 : Tuning in stage 2

1) Put the tuned FLC obtained from Algorithm 1 in parallel with the PD controller.
2) Run the system and obtain L input/output data pairs.
3) Run Algorithm 1 steps 2 and 3.
4) Does the tuned FLC resulting from this algorithm achieve the desired performance.
   - If YES then end the algorithm.
   - If NO then go to the next step.
5) Put the tuned FLC obtained from this algorithm in parallel with the PD controller again.
6) Go to step 2.

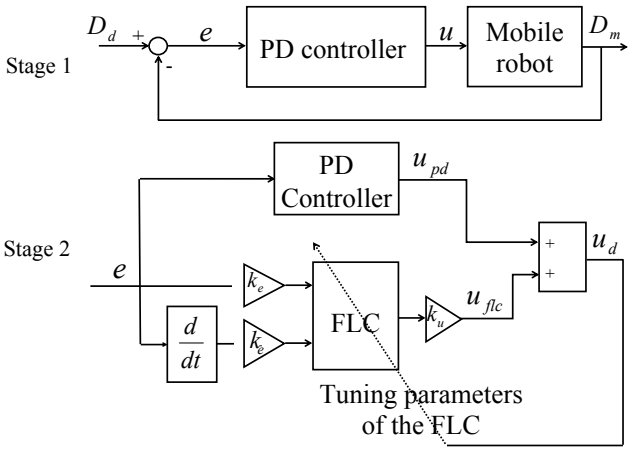

Fig. 5. The proposed GBFLC

In our proposed GBFLC, we construct 3 chromosomes. The first one represents the MF parameters. The second one represents the scaling factors and the third one represents the fuzzy rules' weights.

The proposed GBFLC has 3 variables: 2 inputs, e and $\dot{e}$, and one output, u. Each variable has 5 MFs with the linguistic values HN, LN, Z, LP, and HP. Each MF has 4 coefficients (a, b, c, and d) so the total number of MF parameters to be tuned is 3 variables $\times$ 5 MFs $\times$ 4 coefficients $=$ 60 parameters. So, the chromosome of a MF has a length of 60 genes. We use decimal numbers for coding these parameters. In order to choose consistent values for the MF parameters, we redefine the MF parameters as shown in Fig. 6 to be

$$a = x_o + p_1;$$
$$b = x_o + p_1 + p_2;$$
$$c = x_o + p_1 + p_2 + p_3;$$
$$d = x_o + p_1 + p_2 + p_3 + p_4.$$

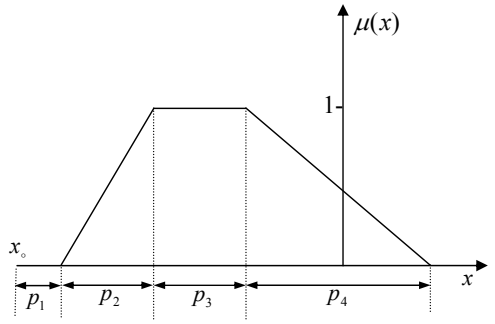

Fig. 6. Trapezoidal MF with modified coefficients

The proposed GBFLC has 3 scaling factors: 2 for the inputs, k_e and $k_{\dot{e}}$ and one for the output, k_u. The chromosome of the scaling factors has 3 genes. We use decimal numbers for coding scaling factors.

Reducing the number of fuzzy rules is important for fuzzy control of complex processes with high dimensionality [14]. The proposed GBFLC has 5 MFs for each input so the total number of fuzzy rules is $5 \times 5 = 25$ fuzzy rules. We tune the weight of each fuzzy rule so, we generate a chromosome of 25 genes which we put in a binary string. When a gene = 0 that means the related fuzzy rule will be omitted and a gene of value 1 means the related fuzzy rule will be added to the rule base.

In our GBFLC, we tried several population sizes. We tried 20, 40, 60, 80, 100, 200, 400, 500, and 1000. We found that a population size of 20 is not enough to get acceptable results. On the other hand, a population size greater than 40 does not significantly improve performance. Therefore, we chose a population size of 40. That number means that we have 40 chromosomes for the MFs, scaling factors, and fuzzy rules' weights. We also tried different numbers of iterations. We tried 10, 15, 20 50, and 100. We found that 10 iterations are not enough to get acceptable results. On the other hand, more than 15 iterations did not significantly improve performance. Therefore, we chose 15 iterations. We used a pentium 4 computer with a 3.2 GHz clock frequency and 1.0 Gigabytes of RAM. The GA program with a population size of 1000 and with 100 iterations took about 2 hours to compute the results and this is computationally expensive and time consuming but a population size of 40 with 15 iterations took about 5 minutes. The probability of crossover is chosen to be 0.4 as such we reproduce 16 chromosomes by crossover. In binary coding, we select a random point as mentioned before, and make the crossover process. In decimal coding, we simply multiply the first chromosome of the selected portion of the old population (20 chromosomes) by a random number r, where $r \in [0,1]$, and multiply the second chromosome by $(1-r)$ then add both of them to generate a new chromosome. We repeat this operation for the next chromosome in the old population until the number of crossover chromosomes is reached. Here, we set $r = 0.5$. The probability of mutation is chosen to be 0.1, therefore, we reproduce 4 chromosomes on average by mutation. For the binary and decimal coding, we generate 4 new chromosomes randomly to avoid a local minimum for the fitness function.

V. Results

In stage 1, we obtain the input/output data pairs from the PD controller and use them to tune our proposed GBFLC as we described in Algorithm 1. We use the PD controller shown in Fig. 2 to teach the FLC; the PD controller is thought of as the initial expert. Fig. 7 shows error resulting from the PD controller versus error resulting from the FLC obtained from stage 1.

We put the tuned FLC obtained from stage 1 in parallel with the PD controller and get new input/output data pairs. We use these data pairs to further tune the FLC obtained from stage 1 as we described in Algorithm 2. We repeated this cycle twice.

Fig. 8 shows error resulting from the PD controller versus error resulting from the PD controller in parallel with the FLC obtained from stage 1. Fig. 9 shows error resulting from the PD controller versus error resulting from the FLC obtained from cycle 1 of stage 2. Fig. 10 shows error resulting from the PD controller versus error resulting from the PD controller in parallel with the FLC obtained from cycle 1 of stage 2. Fig. 11 shows error resulting from the PD controller versus error resulting from the FLC obtained from cycle 2 of stage 2. Fig. 12 shows error resulting from the PD controller versus error resulting from the PD controller in parallel with the FLC obtained from cycle 2 of stage 2.

Fig. 7, Fig. 9, and Fig. 11 show us how the performance of the FLC improves from the first to the second stage (the two cycles) and the best performance is that of the FLC obtained from cycle 2 of stage 2. Fig. 13 shows the paths of the mobile robot to follow a wall using the PD controller and the proposed GBFLC. Fig. 11 and Fig. 13 tell us that our proposed GBFLC is better than the PD controller.

Table II, Table III, and Table IV represent the tuned parameters of the MFs for the input e, the input $\dot{e}$, and the output u, respectively. Table V represents the scaling factors after tuning. Table VI represents the fuzzy decision table after the tuning process. The empty places in Table VI are the omitted rules due to tuning. Here we see that the number of fuzzy rules is reduced by 44% from 25 rules before tuning to 14 after tuning.

TABLE II

Tuned MF parameters for the input e

	a	b	c	d
HN	-1.1277	-0.7744	-0.4998	-0.2049
LN	-0.6312	-0.4160	-0.0660	0.2497
Z	-0.4211	-0.1129	0.1350	0.3521
LP	-0.2460	0.1331	0.4266	0.7748
HP	0.2579	0.6270	0.8409	1.1814

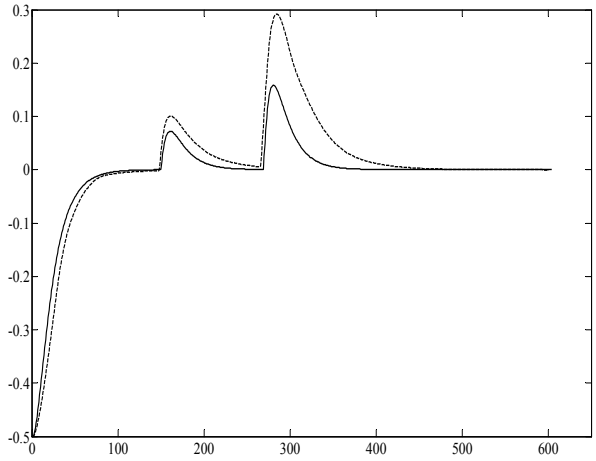

Fig. 7. Error resulting from the PD controller (solid line) vs. error resulting from the FLC obtained from stage 1 (dotted line)

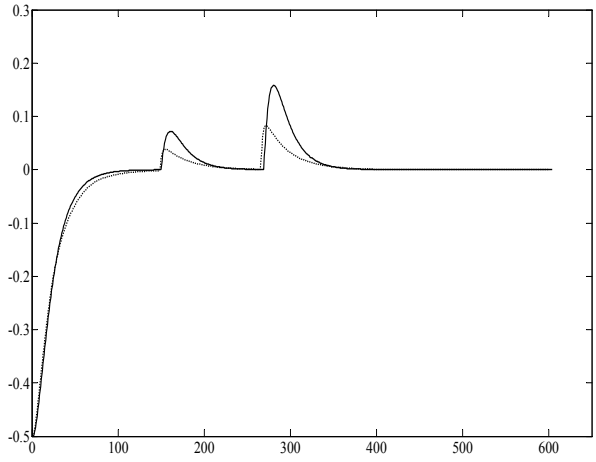

Fig. 8. Error resulting from the PD controller (solid line) vs. error resulting from the PD controller parallel with the FLC obtained from stage 1 (dotted line)

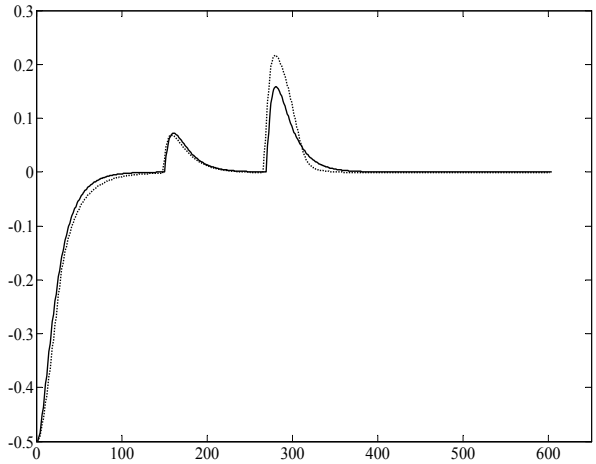

Fig. 9. Error resulting from the PD controller (solid line) vs. error resulting from the FLC obtained from cycle 1 of stage 2 (dotted line)

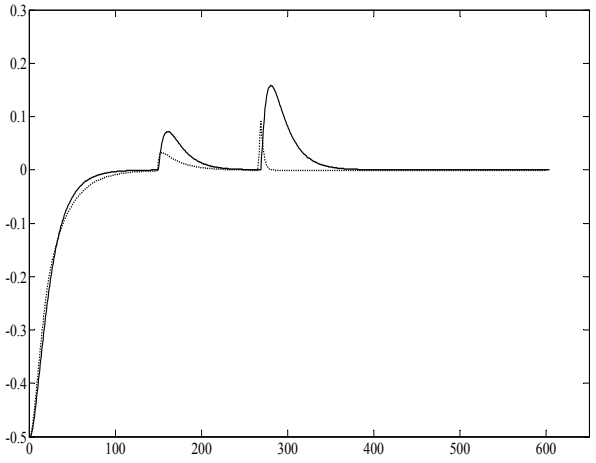

Fig. 10. Error resulting from the PD controller (solid line) vs. error resulting from the PD controller parallel with the FLC obtained from cycle 1 of stage 2 (dotted line)

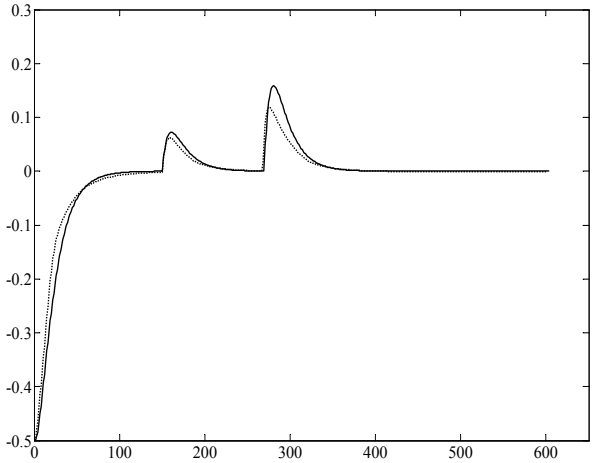

Fig. 11. Error resulting from the PD controller (solid line) vs. error resulting from the FLC obtained from cycle 2 of stage 2 (dotted line)

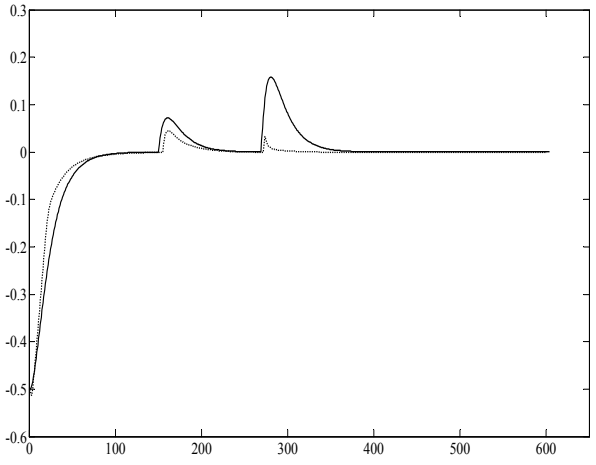

Fig. 12. Error resulting from the PD controller (solid line) vs. error resulting from the PD controller parallel with the FLC obtained from cycle 2 of stage 2 (dotted line)

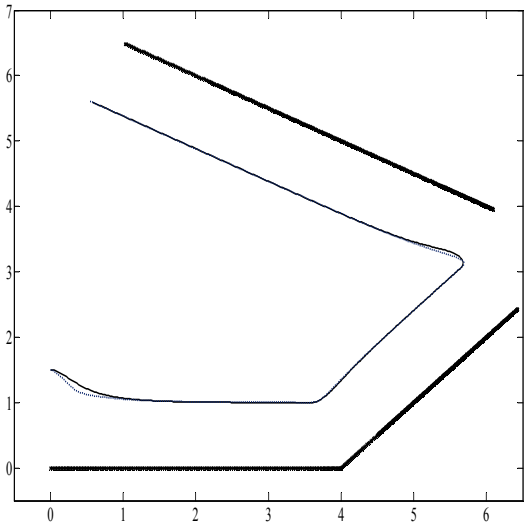

Fig. 13. Paths of the mobile robot to follow a wall using the PD controller (solid line) vs. the GBFLC (dotted line)

TABLE III

TUNED MF PARAMETERS FOR THE INPUT $\dot{e}$

	a	*b*	*c*	*d*
HN	-1.1899	-0.9170	-0.5926	-0.3476
LN	-0.7226	-0.4392	-0.1337	0.2220
Z	-0.4226	-0.1821	0.1041	0.3307
LP	-0.2885	0.0385	0.3351	0.6641
HP	0.3777	0.5834	0.8787	1.1502

TABLE IV

TUNED MF PARAMETERS FOR THE OUTPUT *u*

	a	*b*	*c*	*d*
HN	-1.1782	-0.8579	-0.5208	-0.1903
LN	-0.7220	-0.4580	-0.0858	0.1457
Z	-0.4592	-0.1918	0.1065	0.3961
LP	-0.1679	0.1698	0.4631	0.6976
HP	0.3242	0.6034	0.8046	1.1057

TABLE V

SCALING FACTORS

k_e	$k_{\dot{e}}$	k_u
1.2858	1.6465	1.6206

TABLE VI

FUZZY DECISION TABLE AFTER TUNING

e \ $\dot{e}$	HN	LN	Z	LP	HP
HN		HN	HN		Z
LN		HN	LN	Z	
Z		LN	Z	LP	
LP				HP	HP
HP		LP		HP	HP

VI. CONCLUSIONS

This paper introduces a new technique to tune all the parameters of a FLC using GAs. The proposed GBFLC is used to control a wall-following mobile robot. Two algorithms are described for the tuning process. Algorithm 1 uses a PD controller as the initial expert. Algorithm 2 uses the PD controller and the GBFLC in parallel with to continue improving the performance. In this case, the output of the PD controller is viewed as an error signal that is to be driven to zero. We tune the coefficients of the inputs and the output MFs. Also, we tune the scaling factors of the inputs and the output. Finally, we reduce the number of fuzzy rules. The results show that the performance of the FLC tuned by the proposed technique is better than the performance of the PD controller.

REFERENCES

[1] T. L. Seng, M. B. Khalid, and R. Yusof, "Tuning of a neuro-fuzzy controller by genetic algorithm," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernatics*, vol. 49, April 1999.

[2] T. Fukuda, Y. Hasegawa, K. Shimojima, and F. Saito, "Reinforcement learning method for generating fuzzy controller," in *IEEE International Conference on Evolutionary Computation*, vol. 1, (Perth, WA, Australia), pp. 273–278, 29 November–01 December 1995.

[3] S. Labiod and T. M. Guerra, "Adaptive fuzzy control of a class of SISO nonaffine nonlinear systems," *Fuzzy Sets and Systems*, vol. 158, pp. 1126–1137, 2007.

[4] H. K. Lam and F. H. F. Leung, "Fuzzy controller with stability and performance rules for nonlinear systems," *Fuzzy Sets and Systems*, vol. 158, pp. 147–163, 2007.

[5] M. Mucientes, D. L. Moreno, A. Bugarin, and S. Barro, "Design of a fuzzy controller in mobile robotics using genetic algorithms," *Applied Soft Computing*, vol. 7, pp. 540–546, 2007.

[6] F. Hoffmann, "Evolutionary algorithms for fuzzy control system design," in *Proceedings of the IEEE*, vol. 89, pp. 1318–1333, September 2001.

[7] T. L. Seng, M. Khalid, and R. Yusof, "Tuning of a neuro-fuzzy controller by genetic algorithms with an application to a coupled-tank liquid-level control system," *International Journal of Engineering Applications on Artificial Intelligence*, vol. 11, pp. 517–529, September 1998. Pergamon Press.

[8] E. E. E. Madbouly, A. A. S. Ibrahim, G. Z. El-far, and M. E. Nassef, "Adaptive fuzzy controller using on-line genetic algorithm," in *IEEE International Conference of Electrical, Electronic and Computer*, pp. 14–18, 5–7 September 2004.

[9] S. H. Lim, T. Furukawa, G. Dissanayake, and H. F. Durrant-Whyte, "A time-optimal control strategy for pursuit-evasion games problems," in *International Conference on Robotics Automation*, (New Orleans, LA), IEEE, April 2004.

[10] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[11] J. H. Holland, "Genetic algorithms and the optimal allocation of trials," *SIAM Journal on Computing*, vol. 2, no. 2, pp. 88–105, 1973.

[12] L. Magdalena and F. Monasterio-Huelin, "A fuzzy logic controller with learning through the evolution of its knowledge base," *International Journal of Approximate Reasoning*, vol. 16, pp. 335–358, April 1997.

[13] Z. Hamavand and H. M. Schwartz, "Trajectory control of robotic manipulators by using a feedback-error-learning neural network," *Robotica*, vol. 13, pp. 449–459, 1995.

[14] N. Xiong and L. Litz, "Reduction of fuzzy control rules by means of premise learning - method and case study," *Fuzzy Sets and Systems*, vol. 132, pp. 217–231, 2002.