Sensors and Materials, Vol. 31, No. 9 (2019) 2735–2751
MYU Tokyo

S & M 1969

Using a Type-2 Neural Fuzzy Controller for Navigation Control of Evolutionary Robots

Cheng-Jian Lin,[1*] Jyun-Yu Jhang,[2] and Kuu-Young Young[2]

[1]Department of Computer Science and Information Engineering, National Chin-Yi University of Technology, Taichung 411, Taiwan
[2]Institute of Electrical and Control Engineering, National Chiao Tung University, Hsinchu 300, Taiwan

(Received February 20, 2019; accepted August 1, 2019)

Keywords: mobile robot, type-2 fuzzy neural controller, whale optimization algorithm, navigation control, wall-following control

In this paper, we present an effective navigation control method for mobile robots in an unknown environment. The proposed behavior manager (BM) switches between two behavioral control patterns, wall-following behavior (WFB) and toward-goal behavior (TGB), on the basis of the relationship between the mobile robot and the unknown environment. A type-2 neural fuzzy controller (T2NFC) with an improved whale optimization algorithm (IWOA) is proposed to provide WFB control and obstacle avoidance for mobile robots. In the WFB learning process, the input signal of a controller is the distance between the wall and the sonar sensors, and its output signal is the speed of two wheels of a mobile robot. A fitness function, which operates on the total distance traveled by the mobile robot, distance from the side wall, angle to the side wall, and moving speed, evaluates the WFB performance of the mobile robot. Experimental results reveal that the proposed IWOA is superior to other methods of WFB and navigation control.

1. Introduction

In recent years, the control of mobile robots has been widely used in several applications for solving many problems, such as exploration of unknown environments, object handling, and navigation.[1–3] Navigation control is crucial in mobile robot-based technologies, and its design has become an indispensable research topic.

It is essential to implement mobile robot obstacle avoidance for navigation control. During the navigation process, the mobile robot receives signals from sonar sensors to avoid obstacles and reach the goal point. Recently, smart robots have been widely used for performing several real applications. The two most common mobile robot controllers used are the fuzzy logic and the neural network controllers. Some researchers have combined the concepts of fuzzy logic and neural network into mobile robot controllers. Juang and Chang[4] and Wai and Lin[5] controlled a mobile robot by conveying the information received from sonar sensors to a fuzzy neural network (FNN) controller. Although the wall-following behavior (WFB) control of the

*Corresponding author: e-mail: cjlin@ncut.edu.tw
https://doi.org/10.18494/SAM.2019.2343

ISSN 0914-4935

mobile robot can be implemented successfully, the performance of WFB is not satisfactory in a real environment. To solve the aforementioned problems, several researchers have adopted a type-2 FNN (T2FNN) to design the controller. Jhang and coworkers[6,7] adopted T2FNN to implement a mobile robot controller in real environment. The designed controller can not only escape the dead cycle in WFB control but also successfully complete the navigation task. Lin *et al*. compared the performance of T1FNN and T2FNN controllers. The results demonstrate that the T2FNN controller has better robustness than the T1FNN controller.[8]

T2FNN is an extension of T1FNN that was developed to address the shortcomings of T1FNN. Unlike T1FNN, which uses crisp sets as membership values, T2FNN uses fuzzy sets as membership values. The membership values of these fuzzy sets provide a footprint of uncertainty (FOU), which makes it possible to deal with uncertainties. T2FNN is more efficient than T1FNN in normal environments; however, the number of computations is much higher for T2FNN than for T1FNN. Therefore, interval T2FNN reduces computational complexity and uses the centers of sets (COS) to simplify operations.[9] The related parameters of interval T2FNN are usually adjusted by the backpropagation (BP) algorithm. The BP algorithm is based on a fast gradient descent, which is used to adjust the parameters of interval T2FNN by the error function. The BP algorithm has a high convergence speed. Thus, it is usually used to solve several engineering problems. However, the BP algorithm still has the drawback of being easily trapped into a local minimum. Therefore, many researchers adopted evolutionary computation methods to obtain a global optimum.

Evolutionary algorithms based on the collective behavior of social animals converge rapidly and are easy to implement, but they sometimes perform poorly by demonstrating low accuracy or trapped into local optima. Some evolutionary algorithms are widely used for adjusting the parameters of a neural network or FNN; these evolutionary algorithms include the artificial bee colony (ABC),[10] difference evolution (DE),[11] particle swarm optimization (PSO),[12] and whale optimization algorithm (WOA).[13] Recently, some scholars have applied evolutionary algorithms to develop a controller. Chou and Juang [14] and Lin *et al*.[15] realized mobile robot controllers by using evolutionary algorithms. These controllers[14,15] adopted reinforcement learning and do not need to collect training data in advance. In this study, we focus on WOA, which is based on the hunting strategies of humpback whales, namely, searching for prey, encircling strategy, and the mass net searching strategy. Its advantages are its fast convergence and simple implementation, but it has some shortcomings. For example, it easily falls into a local minimum solution in complex applications. Therefore, an improved whale optimization algorithm (IWOA) is proposed in this study to improve the traditional WOA.

In this paper, we propose a control method for the effective navigation of a mobile robot in unknown environments. To improve the robot's exploration process, the behavior manager (BM) automatically switches between the toward-goal behavior (TGB) and WFB on the basis of the relative positions of the mobile robot and the goal point. In contrast to other methods, we used IWOA, which improved the search capability and enhanced the convergence speed of traditional WOA to adjust a type-2 neural fuzzy controller (T2NFC), and the COS to simplify operations.

The remainder of the paper is organized as follows. In Sect. 2, we introduce the T2NFC based on IWOA. In Sect. 3, we describe the WFB of a mobile robot. Experimental results of navigation control are shown in Sect. 4. In Sect. 5, we present the conclusions.

2. T2NFC Based on Evolutionary Algorithm

In this section, the proposed T2NFC is introduced. An IWOA is also proposed to adjust the parameters of T2NFC. Whereas traditional WOA methods tend to fall into local optima, the proposed IWOA overcomes that tendency.

2.1 T2NFC

Here, the structure of the T2NFC is introduced. Figure 1 shows the five-layer structure of a T2NFC. It consists of an input layer, a fuzzification layer, a firing layer, an output processing layer, and an output layer. The IF-THEN rule can be expressed as

$$R_j\text{: IF } x_1 \text{ is } \tilde{A}_1^j \text{ and } x_2 \text{ is } \tilde{A}_2^j \ldots \text{ and } x_n \text{ is } \tilde{A}_n^j$$
$$\text{THEN } y \text{ is } w_0^j + \sum_{i=1}^{n} w_i^j x_i,$$

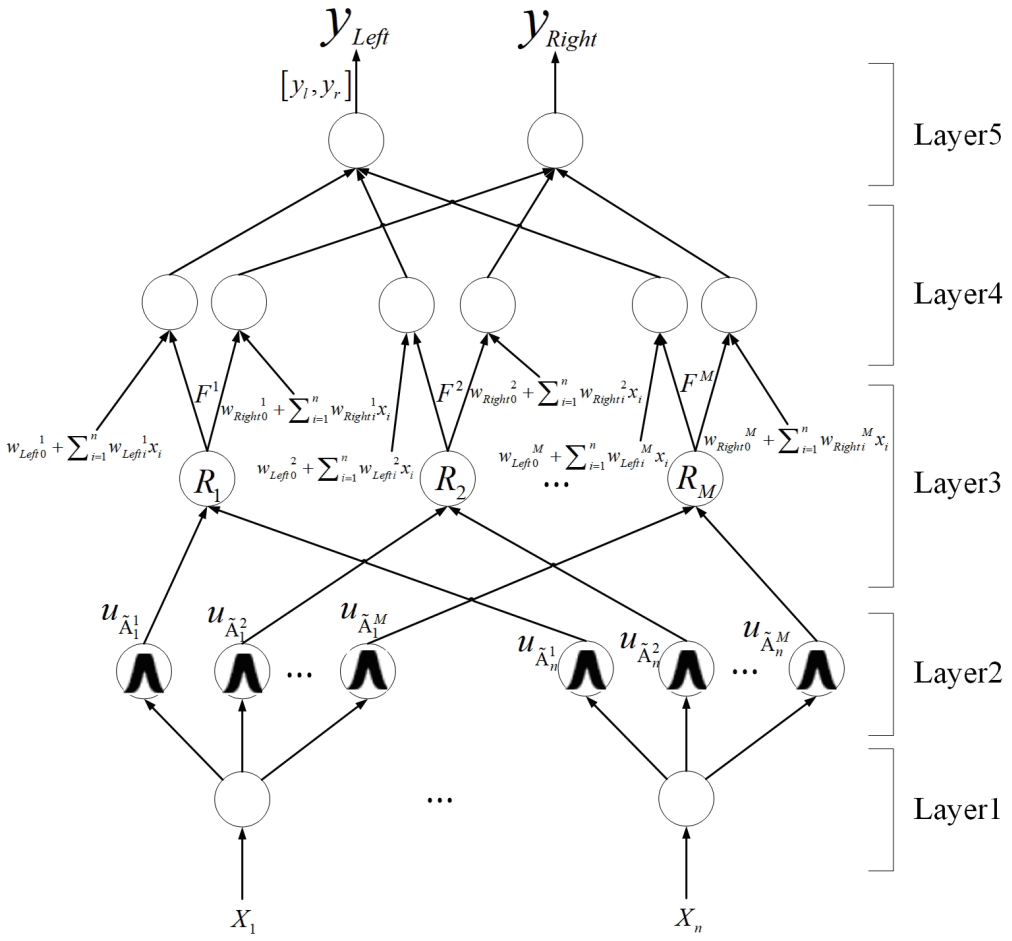


Fig. 1. Structure of a T2NFC.

where j = 1, 2, ..., M is the rule number; i = 1, 2, ..., n is the input number; x_1, x_2, ..., x_n, represents the inputs; $\tilde{A}_1^j, \tilde{A}_2^j, \ldots, \tilde{A}_n^j$ represents the type-2 fuzzy sets; and $w_0^j + \sum_{i=1}^{n} w_i^j x_i$ represents a Takagi–Sugeno–Kang (TSK)-type linear function in the subsequent layer.

Layer 1 only imports the input data into the next layer. Layer 2 performs the fuzzification operation. Each node defines a type-2 fuzzy set. The Gaussian primary membership function has the uncertainty mean [m_1, m_2] and deviation σ. Therefore, the Gaussian primary membership function $u_{\tilde{A}}$ is defined as

$$u_{\tilde{A}_i^j} = \exp\left(-\frac{1}{2} \frac{\left[x_i - m_i^j \right]^2}{\left(\sigma_i^j \right)^2} \right) \equiv N\left(m_i^j, \sigma_i^j; x_i \right), \; m_i^j \in \left[m_{i1}^j, m_{i2}^j \right]. \tag{1}$$

The membership degree of the Gaussian primary membership function $u_{\tilde{A}}$ is the FOU and is expressed as the upper bound $\overline{u}_{\tilde{A}}$ and the lower bound $\underline{u}_{\tilde{A}}$.

$$\overline{u}_{\tilde{A}_i^j} = \begin{cases} N\left(m_{i1}^j, \sigma_i^j; x_i \right), & x_i < m_{i1}^j \\ 1, & m_{i1}^j \le x_i \le m_{i2}^j \\ N\left(m_{i2}^j, \sigma_i^j; x_i \right), & x_i > m_{i2}^j \end{cases} \tag{2}$$

and

$$\underline{u}_{\tilde{A}_i^j} = \begin{cases} N\left(m_{i2}^j, \sigma_i^j; x_i \right), & x_i \le \dfrac{m_{i1}^j + m_{i2}^j}{2} \\ N\left(m_{i1}^j, \sigma_i^j; x_i \right), & x_i > \dfrac{m_{i1}^j + m_{i2}^j}{2} \end{cases} \tag{3}$$

The output of each node is represented as an interval $[\overline{u}_{\tilde{A}_i^j}, \underline{u}_{\tilde{A}_i^j}]$. In layer 3, each node is a rule node, which uses an algebraic product operation to achieve a fuzzy AND operation. The rule node is defined as

$$F^j = \left[\underline{f}^j, \overline{f}^j \right], \tag{4}$$

$$\overline{f}^j = \prod_i^n \overline{u}_{\tilde{A}_i^j} \text{ and } \underline{f}^j = \prod_i^n \underline{u}_{\tilde{A}_i^j}. \tag{5}$$

In layer 3, type-2 fuzzy sets are reduced to type-1 fuzzy sets through a type-reduction operation. The crisp output value [y_l, y_r] is obtained using a center-of-gravity defuzzification method. In this study, we adopted the COS to implement the reduction process; the method is described as

$$y_l = \frac{\sum_{j=1}^{M} \underline{f}^j \left(w_0^j + \sum_{i=1}^{n} w_i^j x_i \right)}{\sum_{j=1}^{M} \underline{f}^j} \tag{6}$$

and

$$y_r = \frac{\sum_{j=1}^{M} \overline{f}^j \left(w_0^j + \sum_{i=1}^{n} w_i^j x_i \right)}{\sum_{j=1}^{M} \overline{f}^j}. \tag{7}$$

Nodes in Layer 5 defuzzify the output by computing the average of y_l and y_r, and the crisp value of y is obtained.

$$y = \frac{y_l + y_r}{2} \tag{8}$$

2.2 Proposed IWOA

In this section, we describe the proposed IWOA. The WOA is based on the hunting behavior of humpback whales, which includes three strategies of encircling prey, bubble-net attacking, and search for prey. The traditional WOA has the advantages of fast convergence and simple implementation, but it easily falls into a local optimal solution in complex applications. Therefore, in this study, the IWOA with Lévy flight strategy is proposed to overcome the shortcomings of the traditional WOA algorithm. The detailed steps of IWOA are as follows:

(1) Coding

All parameters will be encoded into a search agent, which means that each search agent (X) represents a T2NFC, and use IWOA to adjust the parameters. The parameter content contains the uncertainty mean [m_{i1}^j, m_{i2}^j], the deviation σ_i^j, and the consequence weights w_0^j and w_i^j.

(2) Dynamic grouping

Step 1: Calculate the similarity threshold

The fitness values of all search agents X are sorted from high to low, and the group number of all search agents is initialized to 0. The highest fitness value as the leader of the new group and its group number is updated to g where the initial value of g is equal to 1. Then calculate the similarity threshold of this group, which contains the fitness value threshold (ψ) and distance threshold (φ). The value is the average distance difference between the search agents that are not currently grouped (group number 0) and the group leader (τ) and the average fitness value difference. The definition is as follows:

$$DIS^g = \sum_{i=1}^{n} \sum_{j=1}^{D} \sqrt{\left(\tau_j^g - X_j^i \right)^2}, \text{ if } X^i \text{ is ungrouped}, \tag{9}$$

$$FIT^g = \sum_{i=1}^{n} \left| Fit\left(\tau^g\right) - Fit\left(X^i\right) \right|, \text{ if } X^i \text{ is ungrouped,} \tag{10}$$

$$\varphi^g = \frac{DIS^g}{NC}, \tag{11}$$

$$\psi^g = \frac{FIT^g}{NC}, \tag{12}$$

where φ^g and ψ^g represent the distance threshold and fitness value threshold of the gth group, respectively, and τ_j^g is the jth dimension representing the leader of group g; $Fit(\tau^g)$ represents the fitness value of the leader of group g; $Fit(P^i)$ represents the fitness value of the ith search agent, NC denotes the total number of search agents with group number 0 in the current swarm, D represents the dimension of the code, and n is the total number of search agents.

Step 2: Grouping

The ungrouped search agents are sequentially calculated using the following formula to calculate the distance difference (Dis^i) and the fitness value difference (Fit^i) between the self-search agent and the leading search agent.

$$Dis^i = \sum_{j=1}^{D} \sqrt{\left(\tau_j^g - X_j^i\right)^2} \tag{13}$$

$$Fit^i = \left| Fit\left(\tau^g\right) - Fit\left(P^i\right) \right| \tag{14}$$

When $Dis^i < \varphi^g$ and $Fit^i < \psi^g$, indicating that this search agent is similar to the group leader, group them in the same group and update their group number as g; otherwise, the search agent does not belong to the group. If some search agents have not been grouped, repeat steps 1 through 2; conversely, if all agents have their group number, the grouping step is terminated.

(3) Encircling prey

Humpback whales encircle the prey while observing the position of the prey. The best search agent is the current best candidate solution, the other search agents will hence attempt to update their positions towards the best search agent. In addition, the agents will refer to their leader agent τ_g in the group in which they are located. The location update formula is as follows:

$$\vec{D} = \left| \vec{C}.\vec{X}^* - \vec{X}(t) \right| + \left| \vec{X}_{\tau_g} - \vec{X}(t) \right|, \tag{15}$$

$$\vec{X}(t+1) = \vec{X}^*(t) - \vec{A} \cdot \vec{D}, \tag{16}$$

where t is the current iteration, $\vec{A}$ and $\vec{C}$ are the coefficient vectors, $\vec{X}^*$ is the agent position of the best solution, $\vec{X}$ is the agent position, $||$ is the absolute value, and $\cdot$ indicates multiplication.

$\vec{A}$ and $\vec{C}$ are defined as

$$\vec{A} = 2\vec{a} \cdot \vec{r} - \vec{a}\,, \tag{17}$$

$$\vec{C} = 2 \cdot \vec{r}\,, \tag{18}$$

where $\vec{a}$ is linearly decreased from 2 to 0 during the iterations and $\vec{r}$ is a random vector in [0, 1].

(4) Bubble-net attacking method (exploitation phase)

The bubble-net attacking behavior of humpback whales can be divided into the following two strategies:

a: Shrinking encircling operation

The behavior is implemented by decreasing linearly the value of $\vec{a}$ from 2 to 0 in Eq. (17). The value of $\vec{A}$ will be affected by $\vec{a}$, and $\vec{A}$ is a random number in the range [−*a*, *a*]. In this study, $\vec{A}$ is randomly in [−1, 1]. The new position of a search agent can be evaluated anywhere between the original position of the agent and the position of the present best agent.

b: Spiral updating position

First, calculate the distance between the whale and the prey when the humpback whale searches for prey. Then, the humpback whale preys on fish herds and update its position through a helix-shaped movement. The definition is as follows:

$$\vec{X}(t+1) = \vec{D}' \cdot e^{bl} \cdot \cos(2\pi l) + \vec{X}^*(t)\,, \tag{19}$$

$$\vec{D}' = \left|\vec{X}^*(t) - \vec{X}(t)\right|, \tag{20}$$

where $\vec{D}'$ represents the distance from the *i*th whale to the prey, *b* is a constant for defining the logarithmic spiral shapes, and *l* is a random number in [−1, 1].

In fact, the humpback whale swims around the prey in a circle and moves to the prey in a spiral motion at the same time. We assume that humpback whales follow Eqs. (16) and (19) to update the position with a 50% probability, and the definition is as follows:

$$\vec{X}(t+1) = \begin{cases} \vec{X}^*(t) - \vec{A} \cdot \vec{D}, & if\ p < 0.5 \\ \vec{D}' \cdot e^{bl} \cdot \cos(2\pi l) + \vec{X}^*(t), & if\ p > 0.5 \end{cases} \tag{21}$$

where *p* is a random number in the range [0, 1].

(5) Search for prey (exploration phase)

Except for the bubble-net attacking method, humpback whales will randomly search for prey according to the position of each other. Therefore, the $\vec{A}$ vector is used to force a search agent to move far away from a reference whale, and its value is greater than 1 or less than −1. When $\left|\vec{A}\right| > 1$, enforce exploration in the WOA algorithm to perform a global search. Furthermore, to enhance the exploration and avoid the local optimal in WOA, in this study we use the Lévy

flight strategy to improve the shortcomings of traditional WOA. The Lévy flight diversifies search agents and obtains a better trade-off between exploration and exploitation in the WOA. The definition is as follows:

$$\vec{D} = \left| \vec{C}.\vec{X}_{rand} - \vec{X} \right| \oplus \text{Lévy} , \tag{22}$$

$$\vec{X}\left(t+1\right) = \vec{X}_{rand} - \vec{A} \cdot \vec{D} , \tag{23}$$

where $\vec{X}_{rand}$ is a random position vector, and the product $\oplus$ represents entrywise multiplication. The Lévy flight provides a random walk from the given probability distribution and ensures that the search agent could explore the search place efficiently.[16] The definition is as follows.

$$\text{Lévy} \sim u = t^{-\lambda},\ 1 < \lambda \le 3 \tag{24}$$

The generated random step size s of Mantegna's algorithm is used to simulate a λ stable distribution of the Lévy flight[17] as follows:

$$s = \frac{\mu}{\left|\upsilon\right|^{1/\beta}} , \tag{25}$$

$$\sigma_{\mu} = \left[\frac{\Gamma\left(1+\beta\right)\sin\left(\pi\beta/2\right)}{\Gamma\left(\frac{1+\beta}{2}\right)\cdot\beta\cdot 2^{(\beta-1)/2}} \right]^{1/\beta} , \tag{26}$$

where s is the step length of the Lévy flight, which is Lévy(β), β = 1.5 is the Lévy flight exponent, and $\mu = N\left(0,\sigma_{\mu}^{2}\right)$ and $\upsilon = N\left(0,\sigma_{\upsilon}^{2}\right)$ are the normal stochastic distributions with zero means and associated variances, respectively.

3. The Wall-following Control of Mobile Robot

In this section, the proposed T2NFC based on IWOA is demonstrated to control the WFB of a mobile robot. In the proposed method, only the appropriate fitness function is defined to evaluate the performance of mobile robots during the learning process. Designing the control rules by experts and collecting the training data are not necessary.

3.1 Mobile robot description

Figure 2 shows the Pioneer 3-DX mobile robot, which is manufactured by Mobile Robots from the United States. The Pioneer 3-DX is a small lightweight two-wheel two-motor differential drive robot. It is delivered fully assembled with an embedded controller, motors with 500-tick encoders, 19 cm wheels, a tough aluminum body, 8 forward-facing ultrasonic

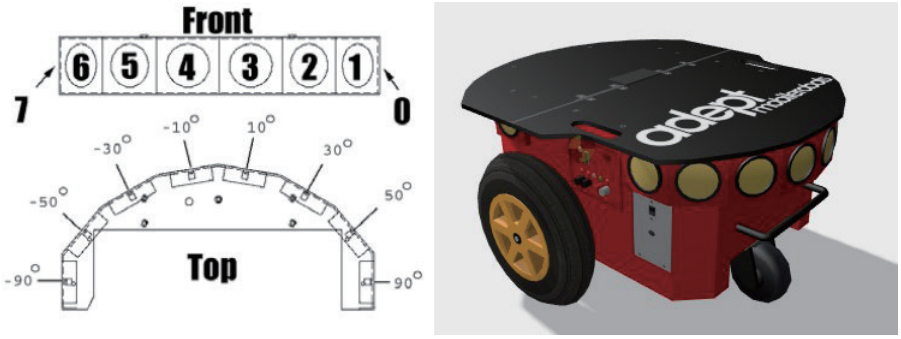


Fig. 2. (Color online) Pioneer3-DX Mobile Robot.

(sonar) sensors, 8 optional rear-facing sonars, 1, 2, or 3 hot-swappable batteries, and a complete software development kit. Users can apply it in a variety of areas and integrate it with all the peripherals to achieve research and development goals.

3.2 Behavior learning of wall-following control

Figure 3 presents a training environment measuring 11 × 8 m^2. To allow mobile robots to encounter different environments, the training environment includes straight lines, corners, and right-angled corners.

The block diagram of WFB control is presented in Fig. 4. Four input signals of the T2NFC, the distances S_1, S_2, S_3, and S_4, are measured by the referred sonar sensors 3, 2, 1, and 0, respectively. Because the arrangement of eight sonar sensors is symmetric, only four sonar sensors on the right side of the robot need to be considered as inputs of the T2NFC during the training process, and the remaining four sonar sensors on the left side of the robot referred to sonar sensors 4, 5, 6, and 7. The detection range of the sonar sensors is limited to 0.1–1 m. The outputs of the T2NFC are the rotational speeds VL and VR of the two wheels. The output ranges from approximately −5.24 to 5.24 rad/s. The execution cycle of a 3-DX mobile robot is 500 ms and is called a time step.

To avoid obstacles during the WFB learning process, three terminal conditions of the mobile robot are defined as follows:

1. The mobile robot is defined to collide with the wall when the measured distance from any sonar sensor is less than 0.1 m.
2. The mobile robot is defined to deviate from the wall when the measured distance of the sensor 0 is greater than 0.7 m.
3. The total moving distance of the mobile robot is larger than the maximum permitted distance of the training environment, namely, 45 m.

In this study, the proposed IWOA was used to train the T2NFC. Whenever any terminal condition of the mobile robot was satisfied, the fitness function was used to evaluate the performance of the mobile robot. The proposed fitness function consists of four subfitness functions, F_1, F_2, F_3, and F_4, which are defined as follows.

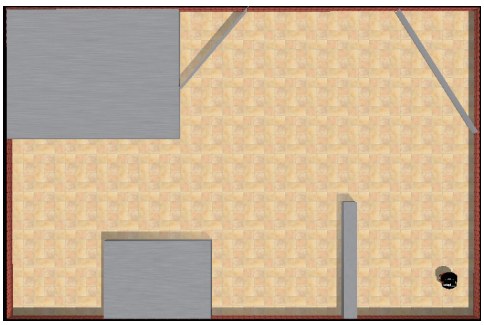


Fig. 3. (Color online) Training environment.

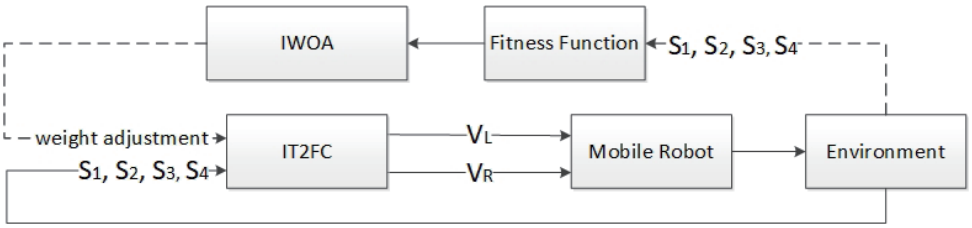


Fig. 4. Block diagram of behavior learning of the wall-following control.

F_1 is used for evaluating the moving distance of the mobile robot. When the moving distance T_{dis} is closer to the predefined value T_{stop}, it indicates that the robot successfully moves around a circular path in the training environment. The subfitness function is defined as

$$F_1 = 1 - \frac{T_{dis}}{T_{stop}} . \tag{27}$$

F_2 is used for evaluating the distance $d_{RW}(t)$ between the side of the robot and the wall. If the robot remains at a fixed distance from the wall, the $d_{RW}(t)$ value is equal to zero and is defined as

$$RD(t) = \left| S_4(t) - d_{wall} \right| , \tag{28}$$

where d_{wall} is a fixed distance (d_{wall} = 0.4 m). The objective function F_{sub2} is defined as the average value of $d_{RW}(t)$ during the moving time

$$F_2 = \frac{\sum_{t=1}^{T_{total}} RD(t)}{T_{total}} . \tag{29}$$

F_3 is used for evaluating the angle θ between the robot and the wall, and $\theta(t)$ is defined as

$$\theta(t) = \cos^{-1}\left(\frac{x(t)^2 + S_4{}^2 - S_3{}^2}{2 \times S_0 \times x(t)} \right), \tag{30}$$

where θ is the angle between the distances S_3 and S_4, and $x(t)$ can be calculated as

$$x(t) = \sqrt{S_4{}^2 + S_3{}^2 - 2S_4S_3\cos 40^\circ}\,. \tag{31}$$

During the movement, to keep the robot parallel to the wall, the objective function F_3 is defined as the average of $|\theta(t) - 90|$ during the total moving time.

$$F_3 = \frac{\sum_{t=1}^{T_{total}} \left|\theta(t) - 90\right|}{T_{total}} \times \frac{1}{90} \tag{32}$$

F_4 is used to evaluate the moving speed of the mobile robot. Therefore, the mobile robot not only maintains a fixed distance from the wall but also increases its moving speed.

$$F_4 = 1 - \frac{V_{avg}}{V_{expt}}, \tag{33}$$

where V_{avg} is the average moving speed of the mobile robot and V_{expt} is the expected speed (set as 0.6 m/s in this study).

A fitness function is formed by combining the four subfitness functions F_1, ..., F_4 and the weighting coefficients α_1, α_2, α_3, and α_4. $F(\cdot)$ is defined as

$$F(\cdot) = \frac{1}{1 + \left(\alpha_1F_1 + \alpha_2F_2 + \alpha_3F_3 + \alpha_4F_4\right)}. \tag{34}$$

The weighting coefficients are set as [α_1, α_2, α_3, α_4] = [0.45, 0.45, 0.05, 0.05]. To avoid the robot collision with obstacles and to keep a fixed distance between the robot and the wall, the two objective functions F_1 and F_2 are very important. Therefore, the weighting coefficients α_1 and α_2 are set to 0.45, whereas the remaining weighting coefficients α_3 and α_4 are set to 0.05.

3.3 Experimental results of wall-following control

In this experiment, to verify the effectiveness of the proposed method, the performance of WFB controlled by the proposed method was compared with the performance of WFB controlled by other methods. Table 1 presents the initial parameters of the IWOA, which are the total population, the Levy(β), the total number of generation, and the number of fuzzy rules. To verify the stability of each algorithm, each method was evaluated 10 times in this experiment.

We compared the proposed method with other evolutionary algorithms.[10–12,14] The proposed IWOA yields the largest fitness value among the algorithms. The comparison results are presented in detail in Table 2. The comparison results comprise the best fitness value, the worst fitness value, the average fitness value, the standard deviation (STD), the number of successful runs, and the required time for moving around a circle. Table 2 shows that the fitness value, the number of successful runs, and the time around a circle of the proposed IWOA are more favorable than those of other algorithms. Figure 5 depicts the trajectories of paths taken

Table 1
Initial parameters of IWOA.

Population	Lévy(β)	Generation	Number of fuzzy rules
30	1.5	3000	6

Table 2
Performance of each algorithm in learning WFB.

Algorithm	Evaluation items					
	Fitness value				Number of successful runs	Time (s)
	Best	Worst	Average	STD		
ABC[10]	0.901	0.891	0.900	0.0043	7	501
DE[11]	0.904	0.894	0.895	0.0069	8	168
PSO[12]	0.915	0.892	0.900	0.0051	9	277
Chou and Juang[14]	0.909	0.894	0.900	0.0046	9	234
IWOA	0.916	0.911	0.914	0.0018	10	127

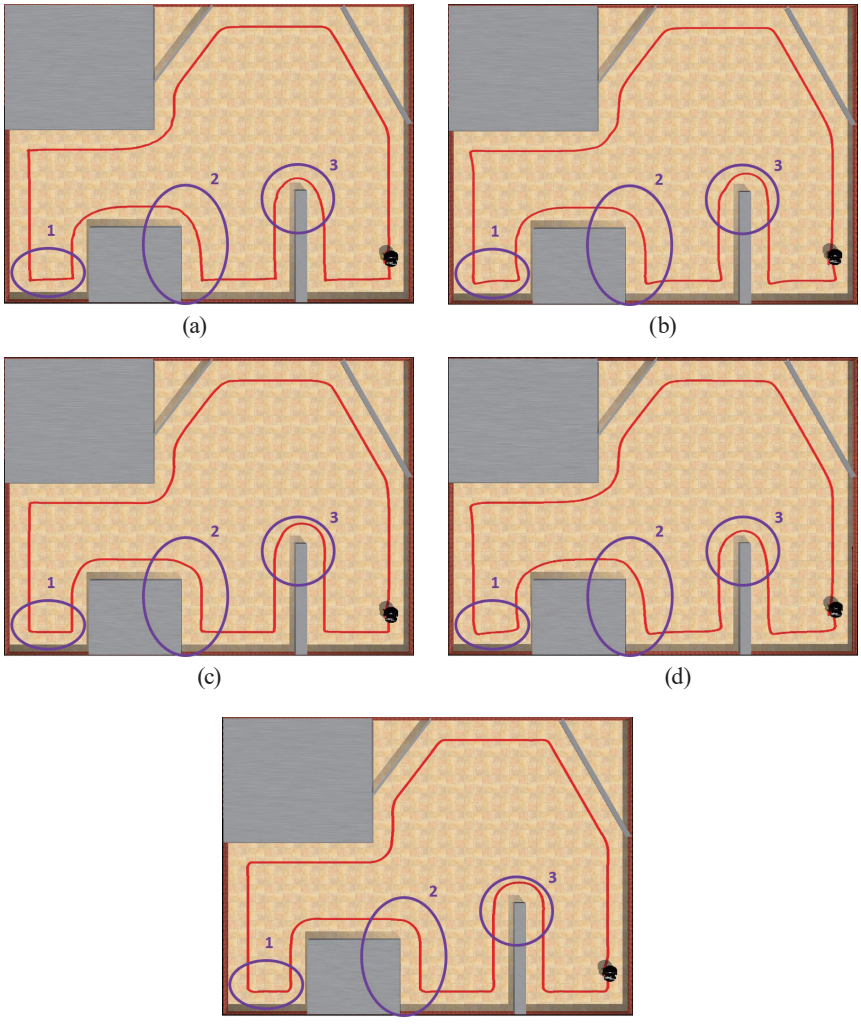


Fig. 5. (Color online) Paths taken by a mobile robot using various algorithms in training environment: (a) ABC, (b) DE, (c) PSO, (d) Chou and Juang,[14] and (e) IWOA.

by a mobile robot, which were obtained using various algorithms in the training environment. In Fig. 5, there are three regions in the training environment, namely, the rectangular groove, right angle bend, and rectangular bump, which are used to evaluate the effectiveness of various controllers. In region 1, the controller using the ABC, PSO, and IWOA methods have a good moving trajectory to pass the rectangular groove and be close to 90 degrees. In region 2, the controller using the PSO and IWOA methods keep a fixed distance from the wall while the mobile robot passes the right angle bend. In region 3, only the controller using the IWOA method has a smoother moving trajectory than the other methods in the rectangular bump.

To verify the WFB control performance of different learning algorithms, unknown test environments were created as shown in Fig. 6. The quantitative evaluation includes the average distance between the robot and the wall (D_{avg}), the moving distance of the robot around a

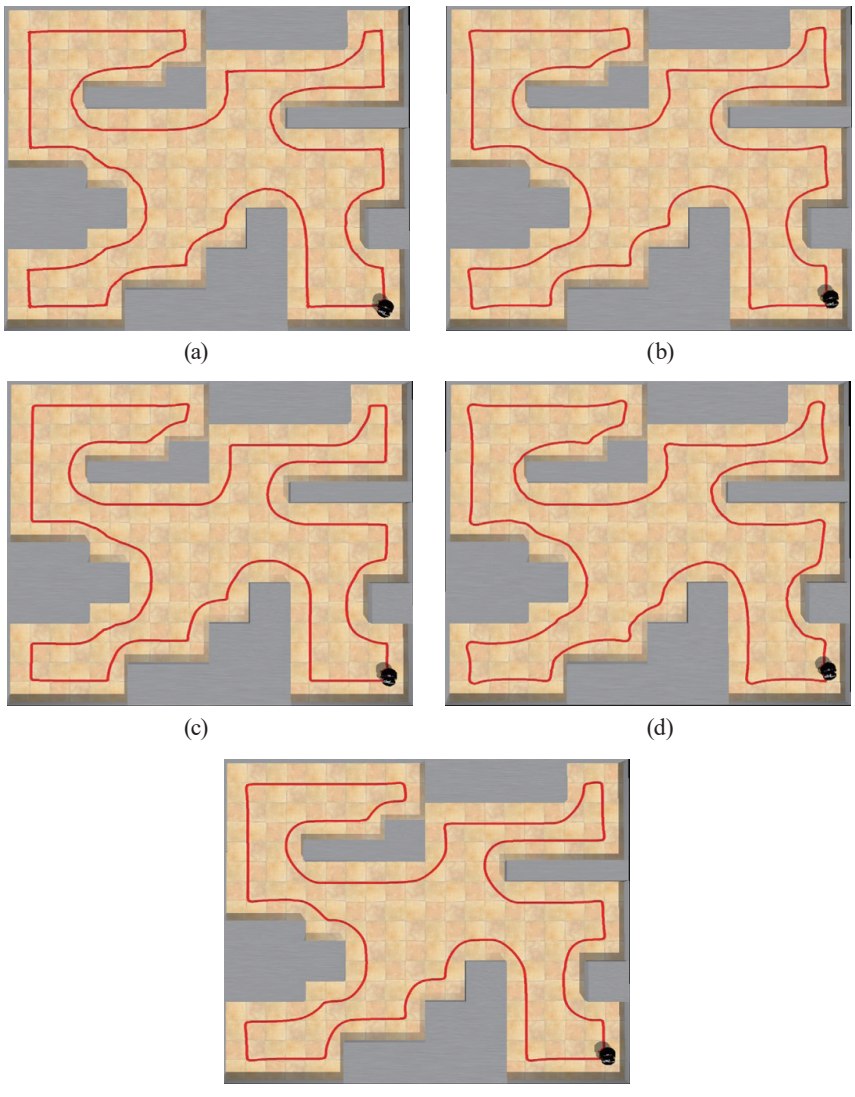


Fig. 6. (Color online) Paths taken by the trained mobile robot in test environments: (a) ABC, (b) DE, (c) PSO, (d) Chou and Juang,[14] and (e) IWOA.

circle (D_{circle}), and the moving time of the robot around a circle (T_{circle}). Table 3 presents the performance of various algorithms in unknown testing environments. The performance of the IWOA algorithm is superior to that of other algorithms. Previously developed methods,[10–12,14] took more time to guide the robot along the wall, such as a circular path in a test environment. The main reason for the different execution times is that the robot needs to keep a fixed distance from the wall when passing the curve. If the controller is designed to work well and the robot is at a fixed distance from the wall, the execution time will be shortened. Otherwise, the robot will swing left and right and will spend more time passing the curve.

4. Wall-following Control of Mobile Robot

In this section, navigation controlled by the proposed method is described. The proposed BM switches the mobile robot behavior according to the relative position between the mobile robot and the goal point.

4.1 BM

The BM has two behavior modes, TGB and WFB. The orientation of the mobile robot toward the goal can be divided into four zones (i.e., R_1, R_2, R_3, and R_4) (Fig. 7). We can determine the location zone (R_n) of the goal point according to the relative position between the mobile robot and the goal point. If the goal point is located at $R_{1–3}$, and an obstacle is detected ($S_i \leq 1$ m), the BM switches to WFB. Otherwise, the BM switches to TGB. If the goal point is located at R_4, the mobile robot will switch to WFB until the robot reaches the goal position.

4.2 Simulation results of navigation control

To verify the performance of the proposed navigation control method, two test environments are used as shown in Fig. 8. Figures 8(a) and 8(b) show four clasp obstacles and a back and forth environment, respectively. Two evaluation factors, the total moving distance (D_{SG}) and total time (T_{SG}) from the start to the goal, were used to analyze the performance of the navigation control. The experimental results are summarized in Table 4, which shows that the proposed method is superior to other methods in terms of D_{SG} and T_{SG}. Therefore, the mobile robot successfully completes the navigation control in unknown environments.

Table 3
Performance of various algorithms in test environments.

Algorithm	Environments		
	Test environment		
	D_{avg} (m)	D_{circle} (m)	T_{circle} (s)
ABC[10]	0.3842	56.8100	385
DE[11]	0.3859	56.4471	239
PSO[12]	0.4050	57.5852	378
Chou and Juang[14]	0.5323	58.9955	297
IWOA	0.3933	56.2204	188

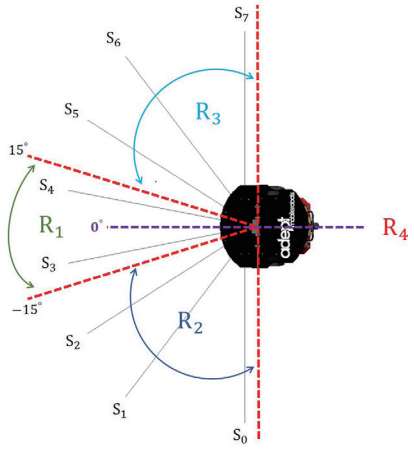
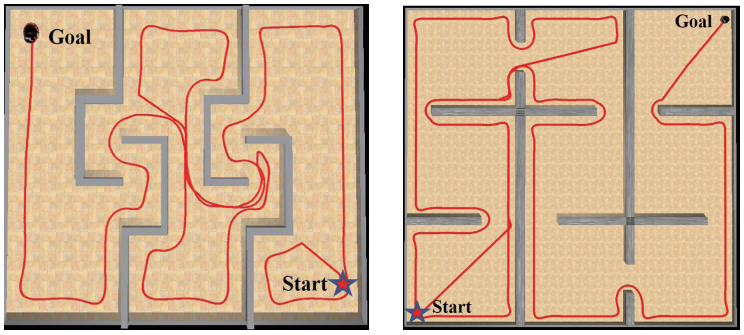


Fig. 7. (Color online) The four orientations of the mobile robot are represented by red dash lines



(a) (b)

Fig. 8. (Color online) Paths taken by a mobile robot obtained using the proposed IWOA algorithm in two test environments: (a) four clasp obstacles and (b) back and forth environment.

Table 4
Comparison results of navigation control using various algorithms.

Environments		Algorithms				
		ABC[10]	DE[11]	PSO[12]	Chou and Juang[14]	IWOA
Fig. 8(a)	DSG (m)	94.77	94.21	95.70	92.21	69.50
	TSG (s)	1051.5	394	616.5	392	240
Fig. 8(b)	DSG (m)	224.32	177.58	178.14	182.49	172.44
	TSG (s)	2092	1188	1257	1125	571

5. Conclusions

In this study, we propose an effective navigation control method in an unknown environment. The proposed BM automatically switches to the WFB or TGB according to the relative position between the mobile robot and the goal point. A T2NFC controller based on the IWOA learning algorithm is used for implementing the WFB of the mobile robot. The proposed IWOA uses the dynamic grouping and Lévy flight strategies to improve the search capability

and the enhanced convergence speed of the traditional WOA. Experimental results reveal that the performance of the proposed method in terms of wall following and navigation control is more efficient than that of other methods in unknown environments.

References

1 L. Tai and M. Liu: 2016 IEEE Int. Conf. Real-time Computing and Robotics (IEEE, 2016). https://doi.org/10.1109/RCAR.2016.7784001
2 J. Chen, M. Gauci, W. Li, A. Kolling, and R. Groß: IEEE Trans. Robot. **31** (2015) 307. https://doi.org/10.1109/TRO.2015.2400731
3 L. Palmieri, A. Rudenko, and K. O. Arras: IEEE Robot. Autom. Lett. **2** (2017) 269. https://doi.org/10.1109/LRA.2016.2602240
4 C. F. Juang and Y. C. Chang: IEEE Trans. Fuzzy Syst. **19** (2011) 379. https://doi.org/10.1109/TFUZZ.2011.2104364
5 R. J. Wai and Y. W. Lin: IEEE Trans. Fuzzy Syst. **21** (2013) 688. https://doi.org/10.1109/TFUZZ.2012.2227974.
6 J. Y. Jhang, C. J. Lin, and K.-Y. Young: Electronics **8** (2019) 298. https://doi.org/10.3390/electronics8030298
7 J. Y. Jhang, C. J. Lin, C. T. Lin, and K.-Y. Young: Int. J. Control Autom. Syst. **16** (2018) 2446. https://doi.org/10.1007/s12555-017-0156-5
8 C. H. Lin, S. H. Wang, and C. J. Lin: Sensors **18** (2018) 4181. https://doi.org/10.3390/s18124181
9 O. Castillo and P. Melin: Appl. Soft Comput. **22** (2011) 1267. https://doi.org/10.1016/j.asoc.2011.12.010
10 Y. Xue, J. Jiang, B. Zhao, and T. Ma: Soft Comput. **22** (2018) 2935. https://doi.org/10.1007/s00500-017-2547-1
11 A. W. Mohamed: J. Intell. Manuf. **29** (2018) 659. https://doi.org/10.1007/s10845-017-1294-6
12 D. Wang, D. Tan, and L. Liu: Soft Computing **22** (2018) 387. https://doi.org/10.1007/s00500-016-2474-6
13 S. Mirjalili and A. Lewis: Adv. Eng. Software **95** (2016) 51. https://doi.org/10.1016/j.advengsoft.2016.01.008
14 C. Y. Chou and C. F. Juang: Inventions **3** (2018) 3. https://doi.org/10.3390/inventions3010003
15 T. C. Lin, C. C. Chen, and C. J. Lin: Adv. Mech. Eng. **10** (2018) 1. https://doi.org/10.1177%2F1687814017752483
16 Y. Ling, Y. Zhou, and Q. Luo: IEEE Access **5** (2017) 6168. https://doi.org/10.1109/ACCESS.2017.2695498
17 M. K. Naik and R. Panda: Appl. Soft Comput. **38** (2016) 661. https://doi.org/10.1016/j.asoc.2015.10.039

About the Authors

Cheng-Jian Lin received his Ph.D. degree in electrical and control engineering from National Chiao-Tung University, Taiwan, R.O.C., in 1996. Currently, he is a distinguished professor of the Computer Science and Information Engineering Department, National Chin-Yi University of Technology, Taichung, Taiwan. His current research interests are machine learning, pattern recognition, intelligent control, image processing, and evolutionary robot. (cjlin@ncut.edu.tw)

Jyun-Yu Jhang received his B.S. and M.S. degrees from Department of Computer Science and Information Engineering, National Chin-Yi University of Technology, Taichung, Taiwan, R.O.C., in 2013 and 2015, respectively. He is currently pursuing his Ph.D, degree at the Institute of Electrical and Control Engineering, National Chiao Tung University, Hsinchu, Taiwan. His current research interests include fuzzy logic theory, type-2 neural fuzzy systems, evolutionary computation, machine learning, and computer vision. (o800920@gmail.com)

Kuu-Young Young received his B.S. degree in electrical engineering from National Taiwan University, Taiwan, R.O.C., in 1983, and his M.S. and Ph.D. degrees in electrical engineering from Northwestern University, Evanston, IL, in 1987 and 1990, respectively. Currently, he is a professor in the Electrical Engineering Department, National Chiao Tung University, Hsinchu, Taiwan. His current research interests are intelligent control, image processing, and robotic control. (kyoung@mail.nctu.edu.tw)