COOPERATION LEARNING FOR BEHAVIOUR-BASED NEURAL-FUZZY CONTROLLER IN ROBOT NAVIGATION

Jianing Li, Jianqiang Yi, Dongbin Zhao and Guangcheng Xi

Institute of Automation, Chinese Academy of Sciences, Beijing 100080, China

Abstract: Based on the previously proposed extended neural-fuzzy network, this paper presents a cooperation scheme of training data based learning and reinforcement learning for constructing sensor-based behaviour modules in robot navigation. In order to solve reinforcement learning problem, a reinforcement-based neural-fuzzy control system (RNFCS) is provided, which consists of a neural-fuzzy controller (NFC) and a neural-fuzzy predictor (NFP). By estimating the "desired output", reinforcement learning is treated from the point of view of training data based learning. Computer simulations are conducted to illustrate the effectiveness of this method.

Keywords: Mobile robots; Behaviour; Neural network; Fuzzy control; Learning algorithms; Sensors

1. INTRODUCTION

The ultimate goal of mobile robot navigation researches is to make robots travelling with high autonomous ability in unknown environments by using on-line sensory information. Since Brooks (1986) proposed the behaviour-based control architecture, this approach has been broadly applied to robot navigation in unknown environments. Unlike the traditional architecture that requires internal representation of the environment, the behaviour-based scheme maps sensor information to control command directly. Fuzzy logic method is an effective solution of representing this mapping relationship as it does not require mathematical model and is able to realize reasoning on uncertain information. However, it is not easy to automatically adjust and improve the performance of a fuzzy system. Bringing the learning abilities of neural networks to fuzzy logic systems may be an effective approach to construct fuzzy systems automatically. Until now, many neural-fuzzy controllers have been developed to automatically design behaviour modules in robot navigation, such as see (Rusu, *et al*., 2003), in which wall following and obstacle avoidance are considered firstly.

According to whether sufficient and consistent training data are available, learning algorithms for behaviour modules can be divided into two kinds: training data based learning such as supervised learning, and non-training data based learning which mainly means reinforcement learning. In general, training data based learning is an effective and fast learning algorithm to automatically construct neural-fuzzy controllers for the behaviour modules such as wall following. But, when it is expensive to obtain sufficient and consistent training data, its capability will degrade greatly. Thus, reinforcement learning requiring no training data seems to be attractive when the learning of non-training data based behaviour modules is considered. However, reinforcement learning usually leads to a heavy learning process. In summary, it is intractable to construct navigation behaviour modules based on neural-fuzzy system by using either training data based learning or non-training data based learning only. Motivated by these observations, selecting wall following and obstacle avoidance as navigation tasks, this paper respectively

This work was partly supported by 973 Project (Grant No. 2003CB517106) and International Cooperation Key Project (Grant No. 2004DFB02100) of Ministry of Science and Technology, China.

constructs wall following behaviour module by using training data based learning and obstacle avoidance behaviour module by using non-training data based learning. So the design of neural-fuzzy controller for behaviour modules can be realized on the cooperation scheme of training data based learning and reinforcement learning. The advantage of cooperation learning is to reduce learning load and make behaviour navigator more suitable in real-time applications.

Based on our previously proposed extended neural-fuzzy network, this paper presents a training data based learning algorithm for structure and parameter learning of the neural-fuzzy controller. For non-training data based learning, enlightened by Sutton and Barto's model (Barto, *et al.*, 1983), this paper proposes a reinforcement-based neural-fuzzy control system (RNFCS), which integrates two neural-fuzzy adaptive elements performing a neural-fuzzy controller (NFC) and a neural-fuzzy predictor (NFP) respectively. Based on our knowledge, there are mainly two kinds of methods for the adjustment of parameters and structure in Sutton and Barto's model-based reinforcement learning: one (Ye, *et al.*, 2003) is originated from Sutton and Barto's model; the other (Lin and Lee, 1994) is based on gradient information. This paper is trying to solve reinforcement learning problem from another point of view. The main idea used is by employing the critic signal and the stochastic exploration method to estimate the "desired output", reinforcement learning problem is changed into training data based learning problem.

2. NEURAL-FUZZY CONTROLLER

2.1 Mobile robot model and the fuzzy controller

The model of the mobile robot used is a cylindrical platform driven by three off-centered omni-directional wheels. Its radius is 0.315m, and is evenly equipped with 18 infrared sensors in a ring. Assuming to work in perfect mode, each sensor covers an angular view of 20^0 and gives a distance to the object in its field of view (0.1m~0.8m). To reduce input dimension, the sensors around the robot are divided into six groups ($S_1 \sim S_6$), each of which consists of three neighboring sensors, as depicted in Fig. 1. Each sensor group is considered as an input variable of the fuzzy controller, and the distance measured by each sensor group is defined as keeping only the smallest value. Considering the omni-directional kinematic nature, the symmetry of the robot, the sensor arrangement, and appropriate number of rules, each input variable is assigned with three fuzzy sets, referring to near, medium and far, respectively. Here, the rotation of the robot is not considered, that means the robot coordinate system is consistent with the world coordinate all the time. Assuming the robot to move with a constant linear velocity, the control variable is defined as the angle from x-axis with 11 fuzzy sets. The initial Gaussian membership functions of the input and output variables are illustrated in Fig. 2.

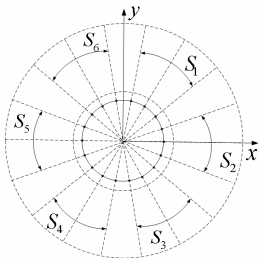


Fig. 1. Mobile robot and sensor arrangement

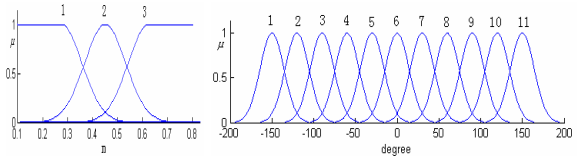


Fig. 2. Membership functions for the input and output variables

In this paper, the following fuzzy if-then rules with certainty grades (Ishibuchi and Nakashima, 2001) are used:

$$R_i: \quad \text{if } x_1 \text{ is } A_{i1} \text{ and } \ldots \text{and if } x_m \text{ is } A_{in}, \quad \text{then } y \text{ is } B_i \text{ with } CF_i$$

where $x = (x_1, \cdots, x_n)$ is a n-dimensional input vector; A_{ij} $(j = 1, \cdots n)$ are antecedent linguistic values; y is a output value; B_i is a consequent linguistic value; CF_i $(0 \le CF_i \le 1)$ is the certainty grade of the fuzzy rule R_i.

2.2 Extended neural-fuzzy controller

This subsection introduces the structure and the function of our previously proposed extended neural-fuzzy controller (NFC) (Li, *et al.*, 2004). Fig. 3 shows the structure of the extended NFC. For convenience, the NFC is considered with two inputs and a single output, expressed by notes in layers 1 and 6, respectively. Nodes in layers 2 and 5 are "term nodes" that act as input and output Gaussian membership functions, respectively. Layer 4 is a normalization layer, which has the same number of nodes as Layer 5. Nodes in layer 3 are "rule nodes", each of which represents the antecedence of a fuzzy rule. The link weights of layers 2 and 3 are set to unity. The widths and centers of output membership functions are viewed as the link weights of layers 5 and 6 respectively. The link weights W_{ij} of layer 4 represent mapping relationships between antecedences and consequences of fuzzy rules, whose strengths express certainty grades CF_{ij} of fuzzy rules, which should be guaranteed by $\sum_{j=1}^{N_3} CF_{ij} = 1$ after learning process. The NFC can work in two manners: down-up and up-down mode, indicated by

real-line and broken arrows, respectively in Fig. 3.

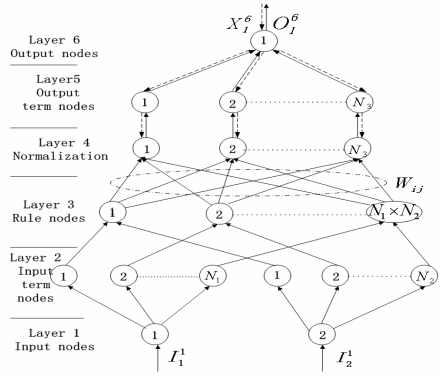


Fig. 3. Extended neural-fuzzy controller

Next, the functions of the nodes in each of the layers are described layer by layer. For convenience, I_i^n and O_i^n are the input and output value of the i th node in layer n, respectively; and m_i^n and σ_i^n are the center and width of the Gaussian function of the i th node in layer n, respectively.

A) Under down-up mode:

Layer 1: $$O_i^1 = \begin{cases} I_1^1, & i = 1,2,\cdots,N_1 \\ I_2^1, & i = N_1+1,\cdots,N_1+N_2 \end{cases} \quad (1)$$

Layer 2: $$I_i^2 = O_i^1 \quad \text{and} \quad O_i^2 = \exp[-(\frac{I_i^2 - m_i^2}{\sigma_i^2})^2] \quad (2)$$

Layer 3: $$I_i^3 = \min(O_j^2, O_k^2) \quad \text{and} \quad O_i^3 = I_i^3 \quad (3)$$

Layer 4: $$I_j^4 = \sum_{i=1}^{N_1 \times N_2} W_{ij} O_i^3 \quad \text{and} \quad O_j^4 = I_j^4 \Big/ \sum_{j=1}^{N_3} I_j^4 \quad (4)$$

Layer 5: $$I_j^5 = O_j^4 \quad \text{and} \quad O_j^5 = \sigma_j^5 I_j^5 \quad (5)$$

Layer 6: $$I_j^6 = O_j^5 \quad \text{and} \quad O_1^6 = \sum_{j=1}^{N_3} m_j^5 I_j^6 \Big/ \sum_{j=1}^{N_3} I_j^6 \quad (6)$$

B) Under up-down mode (X_i^n and Y_i^n are defined as the same as I_i^n and O_i^n respectively):

Layer 6: $$Y_1^6 = X_1^6 \quad (7)$$

Layer 5: $$X_i^5 = Y_1^6 \quad \text{and} \quad Y_i^5 = \exp[-(\frac{X_i^5 - m_i^5}{\sigma_i^5})^2] \quad (8)$$

Layer 4: $$X_i^4 = Y_i^5 \quad (9)$$

Because of the added normalization layer, the benefit of the extended NFC is, when the NFC works under down-up mode, it can avoid imprecise outputs of Layer 4, and then compute the firing strengths of output term nodes more precisely. For realizing behaviour-based navigation tasks, the following sections will introduce the learning process of the extended NFC for setting up fuzzy rules with certainty grades and modifying parameters of membership functions, which is based on training data based hybrid learning or reinforcement learning.

3. TRAINING DATA BASED LEARNING FOR THE NEURAL-FUZZY CONTROLLER

3.1 Initial adjustment of parameters

To narrow the search range of parameter optimization and find fuzzy rules with certainty grades more correctly, Kohonen's self-organized feature mapping learning is used to adjust the centers of membership functions to make them covering only those regions where training data are present. Working under two-sided manner, training input and output data are fed into the NFC from both sides. For each pair of training data, the process of adjusting the centers m_i of membership functions of each input variable is described as follows:

$$\|x(t) - m_{closest}\| = \min_{1 \le i \le k} \{\|x(t) - m_i(t)\|\} \quad (10)$$

$$m_i(t+1) = m_i(t) + \alpha[x(t) - m_i(t)] \quad \text{if } m_i = m_{closest} \quad (11)$$

$$m_i(t+1) = m_i(t) \quad \text{if } m_i \neq m_{closest} \quad (12)$$

where α is a learning rate; and k is the number of input fuzzy sets. The determination of m_{closet} is accomplished via a winner-take-all manner. The similar process is applied simultaneously to adjust the centers of membership functions of the output variable.

3.2 Setting up fuzzy rules with certainty grades

This subsection proposes a new learning algorithm to find fuzzy rules with certainty grades of the NFC that is described as follows.

Step1: Update the link weights. For each incoming training data, the input training data reach the outputs O_i^2 of term nodes in layer 2 under down-up mode, and then get the firing strengths O_i^3 of rules nodes in layer 3. At the same time, the output training data get the inputs X_j^4 of nodes in layer 4. Then the link weighs W_{ij} (set to zero at the start) are updated as:

$$W_{ij}(t+1) = W_{ij}(t) + O_i^3(t) \times X_j^4(t) \quad (13)$$

Step2: Compute the certainty grades. After updating W_{ij} considering all the available training data, the certainty grades are computed as:

$$CF_{ij} = W_{ij} \Big/ \sum_{j=1}^{N_3} W_{ij} \quad (14)$$

which guarantees the sum of certainty grades of fuzzy rules related with each rule node is unity.

3.3 Parameter optimization

After the fuzzy rules with certainty grades have been found, the whole structure of the NFC is established completely, the task of this subsection is to optimize the parameters of membership functions. Here, the gradient descent learning is used. The error function is

$$E = \frac{1}{2}(\tilde{y} - y)^2 \quad (15)$$

where $\tilde{y}$ is the desired output; and y is the current output. Based on input training data, the NFC computes the current output under down-up mode. Then using the error between the current output and the output training data (the desired output), the gradient information is computed layer by layer by chain rule to adjust the parameters of membership

functions. The updated values of the centers and widths of output membership functions are as follows:

$$\Delta m_i^5 = \beta_1(-\frac{\partial E}{\partial m_i^5}) = \eta_1(-\frac{\partial E}{\partial O_1^6}\frac{\partial O_1^6}{\partial m_i^5}) = \beta_1(\tilde{y}-y)\frac{I_i^6}{\sum_{i=1}^{N_3} I_i^6} \quad (16)$$

$$\Delta\sigma_i^5 = \beta_2(-\frac{\partial E}{\partial \sigma_i^5}) = \beta_2(-\frac{\partial E}{\partial O_1^6}\frac{\partial O_1^6}{\partial O_i^5}\frac{\partial O_i^5}{\partial \sigma_i^5})$$

$$= \beta_2(\tilde{y}-y)\left[(m_i^5\sum_{i=1}^{N_3} I_i^6 - \sum_{i=1}^{N_3}(m_i^5 I_i^6))\Big/(\sum_{i=1}^{N_3} I_i^6)^2\right] I_i^5 \quad (17)$$

Similarly, the updating of the centers and widths of input membership functions are by

$$\Delta m_i^2 = \gamma_1\left[-\frac{\partial E}{\partial m_i^2}\right] \text{ and } \Delta\sigma_i^2 = \gamma_2\left[-\frac{\partial E}{\partial \sigma_i^2}\right] \quad (18)$$

where β_1, β_2, γ_1 and γ_2 are learning rates. $\partial E/\partial m_i^2$ and $\partial E/\partial \sigma_i^2$ can be derived by chain rule.

4. REINFORCEMENT LEARNING FOR THE NEURAL-FUZZY CONTROL SYSTEM

4.1 Reinforcement-based neural-fuzzy control system

Enlightened by the Sutton and Barto's model, this paper proposes a reinforcement-based neural-fuzzy control system (RNFCS). The proposed RNFCS, as shown in Fig. 4, integrates two neural-fuzzy adaptive elements into a learning system: one element performing as the NFC and the other as the neural-fuzzy predictor (NFP). The structure of the NFC is based on the extended NFC. The model of the NFP is a four-layer network, which shares the same layers 1and 2 with the NFC, and has individual layers 3 and 4, which are the hidden layer and the output layer with a single node respectively, as shown in Fig. 5. The functions of the nodes in layers 3 and 4 are described as follows.

Layer 3: $I_i^3 = \sum_{j=1}^{N_1+N_2} V_{ij} O_j^2$ and $O_i^3 = \frac{1-\exp(-I_i^3)}{1+\exp(-I_i^3)}$ (19)

Layer 4: $I_1^4 = \sum_{i=1}^{M} U_i O_i^3$ and $O_1^4 = \frac{1-\exp(-I_1^4)}{1+\exp(-I_1^4)}$ (20)

where V_{ij} and U_i are link weights (set to small nonzero values at the start), other symbols are defined as the same as previously.

Here, the working process of the RNFCS is described in brief as follows: at step t, the input vector $x(t)$ supplied by the environment is fed simultaneously into the NFC and the NFP. Based on $x(t)$, the NFP produces a signal $p(t+1)$, which is the prediction of the external reinforcement signal $r(t+1)$ but available at step t; and the NFC gets an output variable $y(t)$. Then using $p(t+1)$ and $y(t)$, the actual output $\hat{y}(t)$ is chosen by the stochastic exploration method introduced in the next subsection. Driven by $\hat{y}(t)$, the system evolves to step $t+1$ and gets $r(t+1)$ by interacting with the environment. Finally, link weights of the NFP are updated by the internal reinforcement signal $\hat{r}(t+1)$, which is the prediction error computed by $r(t+1)$ and $p(t+1)$. $\hat{r}(t+1)$, $y(t)$ and $\hat{y}(t)$ are used for the adjustment of link weights and parameters of the NFC. The learning algorithms will presented in detail in Subsection 4.2 and 4.3.

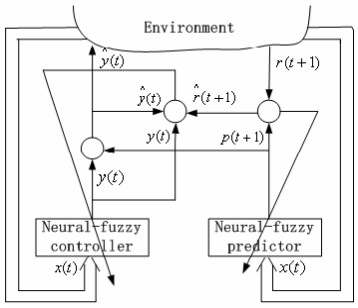


Fig. 4. Reinforcement-based neural-fuzzy control system

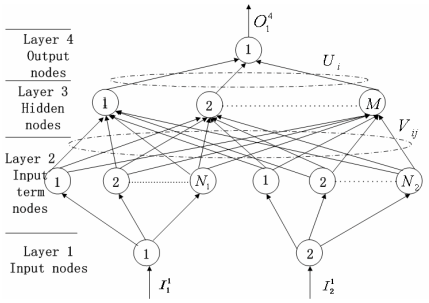


Fig. 5. Neural-fuzzy predictor

4.2 Learning algorithm of the neural-fuzzy predictor

Single-step prediction. Our motivation of designing a RNFCS is for sensor-based obstacle avoidance in un-known environments. The external reinforcement signal r is defined as related with distance measure supplied by sensors. As long as there are obstacles in sensor field of view, r can be obtained at each step, which is only one step behind its corresponding action. So the reinforcement learning problem is simplified to a single-step prediction problem.

In this paper, without using a normal two-valued number for r, which only means "a success" or "a failure", the form of r is defined as a real number, $r \in \{0,1\}$, which represents a detailed and continuous degree of success or failure. r is computed by

$$r = \begin{cases} \frac{1}{0.65}(d_{\min} - 0.15), & if\ 0.15 \le d_{\min} \le 0.8 \\ 0, & if\ 0.1 \le d_{\min} < 0.15 \end{cases} \quad (21)$$

where $d_{\min}$ is the smallest distance measure among sensor groups; and 0.15 is a set threshold for safety.

Learning algorithm of the NFP. The goal of the NFP learning is to adjust link weights of the NFP to minimize $\hat{r}$ at each step, which, based on the single-step prediction, is described as

$$\hat{r} = r - p \quad (22)$$

Here, the gradient descent learning is used to update the link weights of the NFP. The error function is

$$E = \frac{1}{2}(r-p)^2 \tag{23}$$

According to the chain rule, the updated values of the link weights of the NFC are as follows:

$$\Delta U_i = \eta_1(-\frac{\partial E}{\partial U_i}) = \eta_1(r-p)\left[\frac{2\exp(-I_1^4)}{(1+\exp(-I_1^4))^2} O_i^3\right] \tag{24}$$

$$\Delta V_{ij} = \eta_2(-\frac{\partial E}{\partial V_{ij}}) = \eta_2(r-p)\left[\frac{2\exp(-I_1^4)U_i}{(1+\exp(-I_1^4))^2} \frac{2\exp(-I_i^3)O_j^2}{(1+\exp(-I_i^3))^2}\right] \tag{25}$$

where η_1 and η_2 are learning rates.

Stochastic exploration method. When $y(t)$ is produced by the NFC, the conflict between the desire to use $y(t)$ and the desire to further explore the environment to improve the performance of the NFC has to be faced. This paper uses the stochastic exploration method (Lin and Lee, 1994) to overcome this problem. It is described as follows.

Step 1: The range of stochastic exploration is determined by

$$\sigma(t) = \frac{K}{1+e^{A p(t+1)}} \tag{26}$$

where K and A are search-range scaling parameters.

Step 2: A Gaussian random variable $\hat{y}(t)$ is chosen by exploring the $\sigma(t)$ around the mean point $y(t)$.

$\sigma(t)$ can be interpreted as the extent to which the output variable searches for a better action. $p(t+1)$ is the prediction of $r(t+1)$. When $p(t+1)$ is small, $\sigma(t)$ will be large, which means to broaden the search range about the mean $y(t)$. This can provide a higher probability to choose a $\hat{y}(t)$, which is far from $y(t)$, since $y(t)$ is regarded to be far from the best action possible for the current input vector. The similar analysis is true when $p(t+1)$ is large. By using the `above` method, the NFC explores actions possible, and then a better output will be rewarded and a worse one be punished by the learning algorithm of the NFC introduced in the next subsection.

4.3 Learning algorithm of the neural-fuzzy controller

The goal of the NFC learning is to adjust link weights and parameters of the NFC to maximize r at each step, which means to produce an optimal action for each input vector. Basically, the difference between training data based learning and reinforcement learning is: for each input vector, the former can get the instructive signal, which is described as the desired output; and the later has only the critic signal, which represents a reward or a penalty for the output action. If the "desired output" can be produced by employing the critic signal, then a reinforcement learning problem can be changed into a training data based learning problem. In this paper, the learning of the reinforcement-based NFC is considered and solved just based on this idea.

Firstly, using the method proposed by Lin and Lin (1996), the desired output is estimated by

$$y_d(t) \approx y(t) + \rho\frac{\partial r}{\partial y} \quad \text{and} \quad \frac{\partial r}{\partial y} \approx [r-p]_{t+1}\left[\frac{\hat{y}(t)-y(t)}{\sigma(t)}\right] \tag{27}$$

where ρ is a real number, $\rho \in \{0,1\}$; and $y_d(t)$ is the estimated output. If $r(t+1) > p(t+1)$, which means $\hat{y}(t)$ is better than $y(t)$, $\hat{y}(t)$ should be rewarded. So $y_d(t)$ is moved closer $\hat{y}(t)$. On the other side, $y_d(t)$ is moved further away from $\hat{y}(t)$. When the desired output is produced, the reinforcement learning can be changed completely into a training data based learning. The whole process of the training data based hybrid learning in Section 3 can by applied to the learning of the reinforcement-based NFC at each step, in which $(\tilde{y}-y)$ is replaced by $[y_d(t)-y(t)]$.

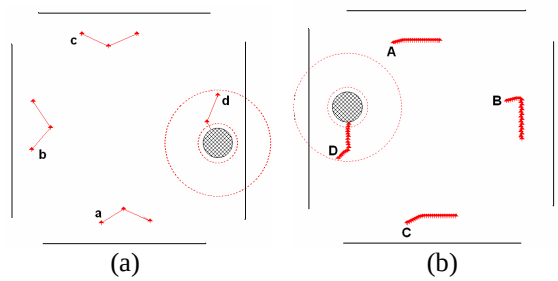
5. SIMULATION AND ANALYSIS

Experiment 1: Wall following learning

In order to obtain training data, the robot is firstly driven by a human operator to move from a, b, c and d start points respectively in a simulated indoor environment as shown in Fig. 6(a). The region between the two dotted circles represents the detectable range of sensors. The folded lines are the trajectories of the robot center. The information supplied by the training data represents that when the robot is close to the wall, it will go further away at next time step; on the contrary, move toward the wall. Parameters used for wall following learning are shown in Table 1. In learning, when any of the six input variables has no measure reading, the third fuzzy set is assumed to be fired with firing strength 1. The velocity of robot is set to 0.05m/step. Driven by the trained NFC, the resulting paths, which start from A, B, C and D respectively, are shown in Fig. 6(b).

Table 1 Parameters used for wall following

α =0.3	η_1 =0.1	η_2 =0.1	η_3 =0.05	η_4 =0.05



(a) (b)

Fig. 6. Training and testing for wall following

Experiment 2: Obstacle avoidance learning

The RNFCS is used for on-line obstacle avoidance in a simulated indoor environment that consists of walls and static obstacles, in which walls also are treated as obstacles, as shown in Fig. 7. At the start, $p(t+1)$, $y(t)$ and $\hat{y}(t)$ are set to zero, and $\sigma(t)$ set to unity. Parameters used for obstacle avoidance learning are shown in Table 2. Driven by the RNFCS, the robot

begins to move from A and B, respectively. When the smallest distance between the robot and obstacles is lower than the given threshold, the robot is backtracked two steps and $p(t+1)$, $y(t)$, $\hat{y}(t)$ and $\sigma(t)$ are set again with initial values. If there are no obstacles in sensor field of view, the robot goes forward one step along x-axis direction. When the robot deals with multiple obstacles, a trap may be encountered, such as around B. The reason is that the robot tends to keep away from all the obstacles. In principle, the robot is always able to get out of the trapped situation by using the stochastic exploration method; however, by adjusting the search-range scaling parameters, the search range can be broadened to speed up learning, and then increase the chance to get out of the trap. At the same time, it should be also noticed that a too large search range will degrade the learning in other situations.

Table 2 Parameters used for obstacle avoidance

$\eta_1 = 0.8$	$\eta_2 = 0.8$	$K = 1$	$A = 1$	$\rho = 1$
$\alpha = 0.6$	$\beta_1 = 0.5$	$\beta_2 = 0.5$	$\gamma_1 = 0.1$	$\gamma_2 = 0.1$

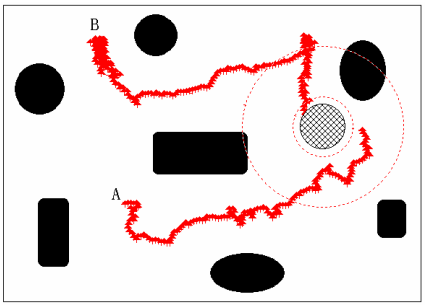


Fig. 7. On-line learning for obstacle avoidance

Experiment 3: Combination navigation

Fig. 8 shows the trajectory of the robot center driven by the combination of wall following and obstacle avoidance in a simulated indoor environment.

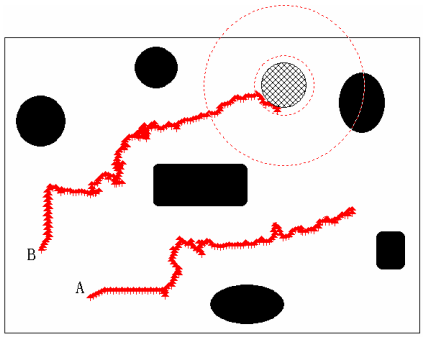


Fig. 8. Combination navigation

A and B are start positions, which are near to walls and there are not obstacles around them. So the robot begins to move under the control of the trained wall following navigation module. This paper does not consider complex coordination strategy for different behaviour modules. When the obstacles are near enough to the robot, the navigation control of the robot is changed under the on-line obstacle avoidance.

6. CONCLUSION

Based on our previously proposed extended NFC, this paper presents a cooperation scheme of training data based learning and non-training data based learning for the design of robot behaviour navigator. According to whether sufficient and consistent training data are available, wall following and obstacle avoidance are designed independently by using training data based hybrid learning and reinforcement learning respectively. Computer simulations show both the behaviour module constructed by off-line training data based learning and the behaviour module built by on-line reinforcement learning work well in unknown environments; while their incorporation lessens learning load of behaviour-based robot navigator and make it more suitable for applications in real-time environments. Future work will focus on the evaluation and improvement of the quality of robot navigation path achieved on the presented method.

REFERENCES

Brooks, R.A. (1986). Robust layered control systems for a mobile robot. *IEEE Transactions on Robotics and Automation.* 2 (1), pp. 14-23.

Rusu, P., E.M. Petriu, T.E. Whalen, A. Cornell and H.J.W. Spoelder (2003). Behavior-based neuro-fuzzy controller for mobile robot navigation. *IEEE Transaction on Instrumentation and Measurement.* 32 (4), pp. 1335-1340.

Ishibuchi, H. and T. Nakashima (2001). Effect of rule weights in fuzzy rule-based classification systems. *IEEE Transaction on Fuzzy Systems.* 9(4), pp. 506-515.

Li, J., J. Yi and D. Zhao (2004). On-line rule generation for robotic behavior controller based on a neural-fuzzy inference network. *Proc. IEEE International Conference on Machine Learning and Cybernetics.* pp: 558-563.

Barto, A.G., R.S. Sutton and C.W. Anderson (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transaction on Systems, Man and Cybernetics.* 13(5), pp. 834-846.

Lin, C. and C.S.G. Lee (1994). Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems. *IEEE Transaction on Fuzzy Systems.* 2(1), pp. 46-63.

Ye, C., N.H.C. Yung and D. Wang (2003). A fuzzy controller with supervised learning assisted reinforcement learning algorithm for obstacle avoidance. *IEEE Transaction on Systems, Man and Cybernetics, Part B.* 33(1), pp. 17-27.

Lin, C. and C. Lin (1996). Reinforcement learning for an ART-based fuzzy adaptive learning control network. *IEEE Transaction on Neural Network.* 7(3), pp. 709-731.