INTERNATIONAL JOURNAL OF ADAPTIVE CONTROL AND SIGNAL PROCESSING
Int. J. Adapt. Control Signal Process. (2011)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/acs.1249

Q(λ)-learning adaptive fuzzy logic controllers for pursuit–evasion differential games

Sameh F. Desouky*, † and Howard M. Schwartz

Department of Systems and Computer Engineering, Carleton University, Ottawa, ON, Canada

SUMMARY

This paper addresses the problem of tuning the input and the output parameters of a fuzzy logic controller. A novel technique that combines Q(λ)-learning with function approximation (fuzzy inference system) is proposed. The system learns autonomously without supervision or *a priori* training data. The proposed technique is applied to three different pursuit–evasion differential games. The proposed technique is compared with the classical control strategy, Q(λ)-learning only, and the technique proposed by Dai *et al.* (2005) in which a neural network is used as a function approximation for Q-learning. Computer simulations show the usefulness of the proposed technique. Copyright © 2011 John Wiley & Sons, Ltd.

Received 28 June 2010; Revised 27 January 2011; Accepted 21 March 2011

KEY WORDS: differential game; function approximation; fuzzy control; pursuit–evasion; Q(λ)-learning; reinforcement learning

1. INTRODUCTION

Fuzzy logic controllers (FLCs) are currently being used 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, 6]. However, FLCs have a drawback of finding its knowledge base which is based on a tedious and unreliable trial and error process. To overcome this drawback, one can use supervised learning [7–11] that needs a teacher or input/output training data. However, in many practical cases the model is totally or partially unknown and it is difficult or expensive and in some cases impossible to get training data. In such cases. It is preferable to use a method such as reinforcement learning (RL).

RL is a computational approach to learning through interaction with the environment [12, 13]. The main advantage of RL is that it does not need either a teacher or a known model. RL is suitable for intelligent robot control especially in the field of autonomous mobile robots [14–18].

Limited studies have applied RL alone to solve environmental problems, but its use with other learning algorithms has increased. In [19], a RL approach is used to tune the parameters of a FLC. This approach is applied to a single case of one robot following another along a straight line. In [15, 20], the authors proposed a hybrid learning approach that combines a neuro-fuzzy system with RL in a two-phase structure applied to an obstacle avoidance mobile robot. In phase 1, supervised learning is used to tune the parameters of an FLC; then in phase 2, RL is employed so that the system can re-adapt to a new environment. The limitation in their approach is that if the training data are hard or expensive to obtain, then supervised learning cannot be applied. In [21],

*Correspondence to: Sameh F. Desouky, Department of Systems and Computer Engineering, Carleton University, Ottawa, ON, Canada.
†E-mail: sameh@sce.carleton.ca, samehfarahat@gmail.com

the authors overcame this limitation by using Q-learning as an expert to obtain training data. Then the training data are used to tune the weights of an artificial neural network (NN) controller applied to a mobile robot path-planning problem.

In this work, we are interested in the pursuit–evasion differential game since it can be considered as a general problem for many other robotic problems such as obstacle avoidance, leader–follower, path-planning and wall-following problems. In addition, it is useful for many real-world applications including surveillance and tracking, search and rescue, locating and capturing hostile and localizing and neutralizing environmental threads. In [22], a multi-robot pursuit–evasion game is investigated. The model consists of a combination of aerial and ground vehicles. However, the unmanned vehicles are not learning. They just do the actions they receive from a central computer system. In [23], a hierarchical software structure for a multi-player pursuit–evasion game is proposed. However, the robots do not learn or adapt to the changing environment. In [24], the use of RL in the multi-agent pursuit–evasion problem is discussed. The individual agents learn a particular pursuit strategy. However, the authors do not use a realistic robot model or robot control structure. In [25], RL is used to tune the output parameters of an FLC in a pursuit–evasion game.

The problem assigned in this paper is that we assume that the pursuer/evader does not know its control strategy. It is not told which actions to take so as to be able to optimize its control strategy. We assume that we do not even have a simplistic PD controller strategy [10, 11]. The learning goal is to make the pursuer/evader able to self-learn its control strategy. It should do that online by interaction with the evader/pursuer.

In this work, we are not interested in deriving the optimal control strategy. We focus on adaptively learning and designing FLCs online when the expert or the training data are not available. From several learning techniques, we choose RL. RL methods learn without a teacher, without anybody telling them how to solve the problem. RL is related to problems where the learning agent does not know what it must do. It is the most appropriate learning technique for our problem.

However, using RL alone has a limitation in that the RL method is designed only for discrete state–action spaces. Since we want to use RL in the robotics domain which is a continuous domain, then we need to use some types of function approximation such as fuzzy inference systems (FISs) to generalize the discrete state–action space into a continuous state–action space. Therefore, from the RL point of view, an FIS is used as a function approximation to compensate for the limitation in RL. And from the FIS point of view, RL is used to tune the input and/or the output parameters of the FIS. In this case, the FIS is used as an adaptive controller whose parameters are tuned online by RL. Therefore, combining RL and FIS has two objectives, to compensate for the limitation in RL and to tune the parameters of the FLC.

A number of papers used FIS as a function approximation with Q-learning [26–28]. However, these works have the following disadvantages: (i) the action space is considered to be discrete and (ii) only the output parameters of the FIS are tuned.

In this paper, we propose a novel technique called Q(λ)-learning fuzzy inference system (QLFIS). The proposed QLFIS, which combines Q(λ)-learning with FIS, is used directly with the continuous state–action spaces. In addition, it is used to tune the input and the output parameters of the FLC. The learning process in the proposed QLFIS is performed simultaneously. The FIS is used as a function approximation to estimate the optimal action-value function, $Q^*(s, a)$, in the continuous state and action space while Q(λ)-learning is used to tune the input and the output parameters of both the FIS and the FLC.

The proposed technique is applied to three different pursuit–evasion differential games. We start with a simple pursuit–evasion game in which only the pursuer self-learns its control strategy online while the evader plays a simple classical control strategy. In the second game, we make the evader play an intelligently control strategy by exploiting the advantage of its higher maneuverability during the game. Finally, we increase the complexity of the system by assuming both the pursuer and the evader do not know their control strategies or the other's control strategy.

This paper is organized as follows: some basic terminologies for RL and FIS are reviewed in Sections 2 and 3, respectively. In Section 4, the pursuit–evasion game is described. The proposed

DOI: 10.1002/acs

QLFIS technique is described in Section 5. Section 6 presents the computer simulation and the results are discussed in Section 7. Finally, conclusion is pointed out in Section 8.

2. REINFORCEMENT LEARNING

Agent–environment interaction in RL is shown in Figure 1 [12]. It consists mainly of two blocks, an agent which tries to take actions so as to maximize a discounted return, R, and an environment which provides the agent with rewards. The discounted return, R_t, at time t is defined as

$$R_t = \sum_{k=0}^{\tau} \gamma^k r_{t+k+1} \tag{1}$$

where r_{t+1} is the immediate reward, γ is the discount factor ($0<\gamma\leqslant 1$) and τ is the terminal point. Any task can be divided into independent episodes and τ is the end of an episode. If τ is finite, then the model is called a finite-horizon model [13]. If $\tau \to \infty$, then the model is called an infinite-horizon discounted model and in this case $\gamma<1$ to avoid infinite total rewards.

The performance of an action, a, taken in a state, s, under policy, π, is evaluated by the action value function, $Q^{\pi}(s,a)$,

$$\begin{aligned} Q^{\pi}(s,a) &= E_{\pi}(R_t | s_t = s, a_t = a) \\ &= E_{\pi}\left(\sum_{k=0}^{\infty} \gamma^k r_{k+t+1} | s_t = s, a_t = a \right) \end{aligned} \tag{2}$$

where $E_{\pi}(\cdot)$ is the expected value under policy, π. The RL method searches for the optimal policy, π^*, by searching for the optimal value function, $Q^*(s,a)$ where

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) \tag{3}$$

Many methods have been proposed for estimating the optimal value functions. Here, we focus on the temporal difference (TD) method. The TD method has several control algorithms. The most widely used and well-known control algorithm is Q-learning [29].

Q-learning, which was first introduced by Watkins in his PhD [30], is an off-policy algorithm. This means that it has the ability to learn without following the current policy. The state and action spaces are discrete and their corresponding value function is stored in what is known as a Q-table. To use Q-learning with continuous systems (continuous state and action spaces), one can discretize the state and action spaces [21, 31–36] or use some type of function approximation such as FISs [26–28], NNs [12, 19, 37], or use some type of optimization technique such as genetic algorithms [38, 39].

A one-step update rule for Q-learning is defined as

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha \triangle_t \tag{4}$$

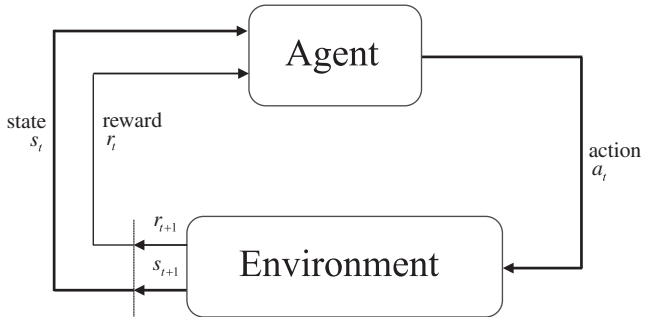

Figure 1. Agent–environment interaction in RL.

DOI: 10.1002/acs

where α is the learning rate ($0<\alpha\leqslant 1$) and $\triangle_t$ is the temporal difference error (TD-error) defined as

$$\triangle_t = r_{t+1} + \gamma \max_{\acute{a}\in A} Q_t(s_{t+1}, \acute{a}) - Q_t(s_t, a_t) \tag{5}$$

where $\acute{a}$ is the action that satisfies the maximum operator. Equation (4) updates the value function according to the immediate reward obtained from the environment. To update the value function based on a multi-step update rule, one can use eligibility traces [12]. Eligibility traces are used to modify a one-step TD algorithm, TD(0), to be a multi-step TD algorithm, TD(λ). The eligibility trace for the continuous state and action spaces is defined as

$$e_t = \gamma\lambda e_{t-1} + \frac{\partial Q_t(s_t, a_t)}{\partial \phi} \tag{6}$$

where $e_0 = 0$, λ is the trace-decay parameter ($0\leqslant\lambda\leqslant 1$) and ϕ is the parameter to be tuned. Eligibility traces are used to speed up the learning process and hence to make it suitable for online applications. Now we will modify (4) to be

$$Q_{t+1}(s, a) = Q_t(s, a) + \alpha\triangle_t e_t \tag{7}$$

3. FUZZY INFERENCE SYSTEM

The most widely used FIS model is Takagi–Sugeno–Kang (TSK) [40]. A first-order TSK means that the output is a linear function of its inputs, whereas a zero-order TSK means that the output is a constant function. In this work, a zero-order TSK model is used.

A Rule used in zero-order TSK models for N inputs has the form

$$R_l: \text{IF } x_1 \text{ is } A_1^l \text{ AND } \ldots \text{ AND } x_N \text{ is } A_N^l \text{ THEN } f_l = K_l \tag{8}$$

where A_i^l is fuzzy set of the ith input variable, x_i, in rule R_l, $l = 1, 2, \ldots, L$ and K^l is the consequent parameter of the output, f_l, in rule R_l. The fuzzy output can be defuzzified into a crisp output using one of the defuzzification techniques. Here, the weighted average method is used and is defined as follows:

$$f(\bar{x}) = \frac{\sum_{l=1}^{L}\left(\prod_{i=1}^{N}\mu^{A_i^l}(x_i)\right)K_l}{\sum_{l=1}^{L}\left(\prod_{i=1}^{N}\mu^{A_i^l}(x_i)\right)} \tag{9}$$

where $\mu^{A_i^l}(x_i)$ is the membership value for the fuzzy set A_i^l of the input x_i in rule R_l. Owing to its simple formulas and computational efficiency, Gaussian membership function (MF) has been used extensively especially in real-time implementation and control. The Gaussian MF is defined as

$$\mu^{A_i^l}(x_i) = \exp\left(-\left(\frac{x_i - m_i^l}{\sigma_i^l}\right)^2\right) \tag{10}$$

where σ and m are the standard deviation and the mean, respectively.

The structure of the FIS used in this work is shown in Figure 2. Without loss of generality, we assume that the FIS model has 2 inputs, x_1 and x_2, and one output, f. Each input has three Gaussian MFs. The structure has two types of nodes. The first type is an adaptive node (a squared shape) whose output needs to be adapted (tuned) and the second type is a fixed node (a circled shape) whose output is a known function of its inputs.

The structure has five layers. In layer 1, all nodes are adaptive. This layer has six outputs denoted by O^1. The output of each node in layer 1 is the membership value of its input defined by (10). In layer 2, all nodes are fixed. The AND operation (multiplication) between the inputs of each rule

DOI: 10.1002/acs

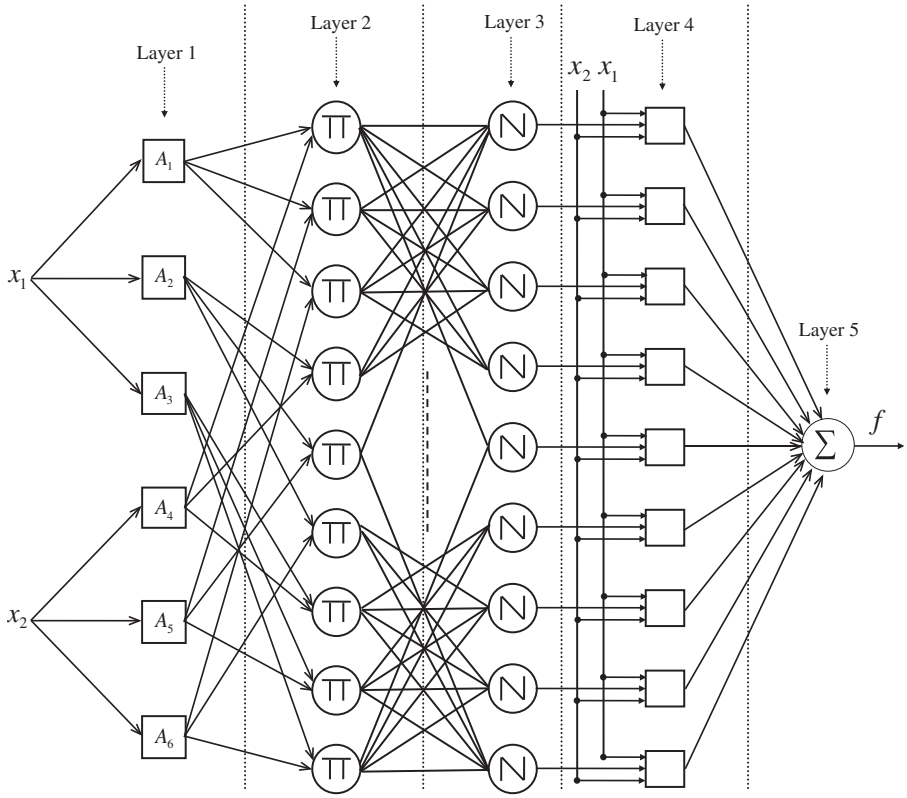

Figure 2. Structure of an FIS model.

is calculated in this layer. This layer has nine outputs denoted by O_l^2, $l=1,2,\ldots,9$. The output of each node in layer 2, ω_l, is known as the firing strength of the rule. It is calculated as follows:

$$O_l^2=\omega_l=\prod_{i=1}^{2}\mu^{A_i^l}(x_i) \tag{11}$$

In layer 3, all nodes are fixed. This layer has nine outputs denoted by O_l^3. The output of each node in layer 3 is the normalized firing strength, $\overline{\omega}_l$, which is calculated as follows:

$$O_l^3=\overline{\omega}_l=\frac{O_l^2}{\sum_{l=1}^{9}O_l^2}=\frac{\omega_l}{\sum_{l=1}^{9}\omega_l} \tag{12}$$

In layer 4, all nodes are adaptive. The defuzzification process is performed in this layer and the next layer. Layer 4 has nine outputs denoted by O_l^4. The output of each node is

$$O_l^4=O_l^3K_l=\overline{\omega}_lK_l \tag{13}$$

Layer 5 is the output layer and has only one fixed node whose output, f, is the sum of all its inputs as follows:

$$O^5=f=\sum_{l=1}^{9}O_l^4=\sum_{l=1}^{9}\overline{\omega}_lK_l \tag{14}$$

which is the same as (9).

DOI: 10.1002/acs

4. PURSUIT–EVASION DIFFERENTIAL GAME

The pursuit–evasion differential game is one application of differential games [41] in which a pursuer tries to catch an evader in minimum time where the evader tries to escape from the pursuer. The pursuit–evasion model is shown in Figure 3. This model is known as the game of two cars [41]. Equations of motion for the pursuer/evader robot are [42]

$$\begin{aligned} \dot{x}_i &= V_i \cos(\theta_i) \\ \dot{y}_i &= V_i \sin(\theta_i) \\ \dot{\theta}_i &= \frac{V_i}{L_i} \tan(u_i) \end{aligned} \tag{15}$$

where 'i' is 'p' for the pursuer and is 'e' for the evader, (x_i, y_i) is the position of the robot, V_i is the velocity, θ_i is the orientation, L_i is the wheelbase, and u_i is the steering angle where $u_i \in [-u_{i_{\max}}, u_{i_{\max}}]$. The minimum turning radius of the robot is calculated as

$$Rd_{i_{\min}} = \frac{L_i}{\tan(u_{i_{\max}})} \tag{16}$$

Our scenario is to make the pursuer faster than the evader ($V_{\mathrm{p}} > V_{\mathrm{e}}$), but at the same time to make it less maneuverable than the evader ($u_{\mathrm{p}_{\max}} < u_{\mathrm{e}_{\max}}$). The classical control strategy that we compare with our results is defined as

$$u_i = \begin{cases} -u_{i_{\max}} & : \quad \delta_i < -u_{i_{\max}} \\ \delta_i & : \quad -u_{i_{\max}} \leqslant \delta_i \leqslant u_{i_{\max}} \\ u_{i_{\max}} & : \quad \delta_i > u_{i_{\max}} \end{cases} \tag{17}$$

where

$$\delta_i = \tan^{-1}\left(\frac{y_{\mathrm{e}} - y_{\mathrm{p}}}{x_{\mathrm{e}} - x_{\mathrm{p}}}\right) - \theta_i \tag{18}$$

The capture occurs when the distance between the pursuer and the evader is less than a certain amount, ℓ. This amount is called the capture radius, which is defined as

$$\ell = \sqrt{(x_{\mathrm{e}} - x_{\mathrm{p}})^2 + (y_{\mathrm{e}} - y_{\mathrm{p}})^2} \tag{19}$$

In the simple pursuit–evasion game described by (17) and (18), the robots are simulated with only knowledge of the angle of the line of sight (LOS) to the other robot. As such, the optimal solution in this case is for each robot to try and make the angle of the LOS zero and we refer

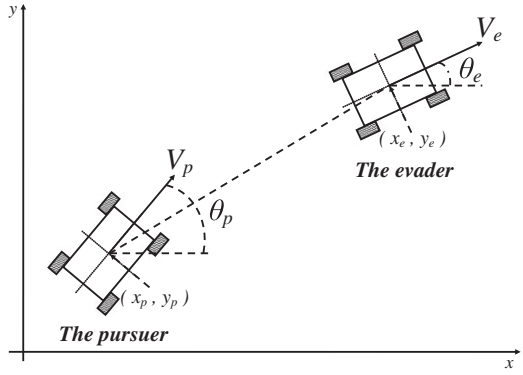

Figure 3. Pursuit–evasion model.

DOI: 10.1002/acs

to this as the 'classical' strategy. In Section 7, we will demonstrate how the learning algorithms will achieve this goal. In Section 7, we will also show the case when the evader has knowledge of closing distance and therefore the evader will turn and take advantage of its higher maneuverability. Once again, we will demonstrate how the pursuer will be able to learn to turn as well.

One reason for choosing the pursuit–evasion game is that the optimal control strategy for a simple game can be determined. Therefore, it can be used as a reference for our results. In this way, we can check the validity of our proposed technique.

5. THE PROPOSED QLFIS

An FIS is used as a function approximation for Q(λ)-learning to generalize the discrete state and action spaces into continuous state and action spaces and at the same time Q(λ)-learning is used to tune the parameters of the FIS and the FLC. The structure of the proposed QLFIS is shown in Figure 4 which is a modified version of the proposed techniques used in [19, 25].

The difference between the proposed QLFIS and that proposed in [25] is that in [25], the authors used FIS to approximate the value function, $V(s)$, but the proposed QLFIS is used to approximate the action–value function, $Q(s,a)$. In addition in [25], the authors tune only the output parameters of the FIS and the FLC, whereas in this work the input and the output parameters of the FIS and the FLC are tuned. The reason for choosing Q-learning in our work is that it outperforms the actor-critic learning [43]. The main advantage of Q-learning over actor-critic learning is exploration insensitivity, since Q-learning is an off-policy algorithm (see Sections 2) whereas actor-critic learning is an on-policy algorithm.

The difference between the proposed QLFIS and that proposed in [19] is that in [19], the authors used NNs as a function approximation, but here we use the FIS as a function approximation. There are some advantages of using FIS rather than NNs such that: (i) linguistic fuzzy rules can be obtained from human experts [44] and (ii) the ability to represent fuzzy and uncertain knowledge [45]. In addition, our results show that the proposed QLFIS outperforms the technique proposed in [19] in both the learning time and the performance.

Now we will derive the adaptation laws for the input and the output parameters of the FIS and the FLC. The adaptation laws will be derived only once and are applied for both the FIS and the FLC. Our objective is to minimize the TD-error, Δ_t, and by using the mean square error we can formulate the error as

$$E = \tfrac{1}{2}\Delta_t^2 \tag{20}$$

We use the gradient descent approach and according to the steepest descent algorithm, we make a change along the negative gradient to minimize the error; hence,

$$\phi(t+1) = \phi(t) - \eta \frac{\partial E}{\partial \phi} \tag{21}$$

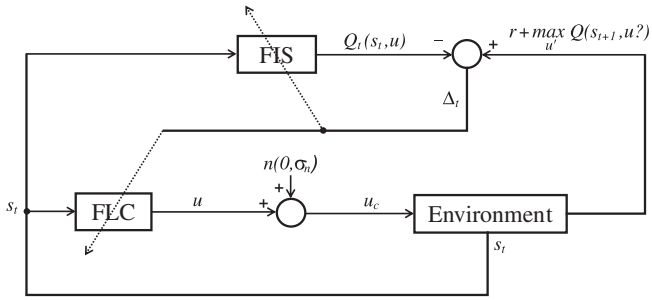

Figure 4. The proposed QLFIS technique.

DOI: 10.1002/acs

where η is the learning rate and ϕ is the parameter vector of the FIS and the FLC, where $\phi=[\sigma, m, K]$. The parameter vector, ϕ, is to be tuned. From (20) we get

$$\frac{\partial E}{\partial \phi}=\Delta_t \frac{\partial \Delta_t}{\partial \phi} \tag{22}$$

Then from (5),

$$\frac{\partial E}{\partial \phi}=-\Delta_t \frac{\partial Q_t(s_t, u_t)}{\partial \phi} \tag{23}$$

Substituting in (21), we get

$$\phi(t+1)=\phi(t)+\eta\Delta_t \frac{\partial Q_t(s_t, u_t)}{\partial \phi} \tag{24}$$

We can obtain $\partial Q_t(s_t, u_t)/\partial\phi$ for the output parameter, K_l, from (14), where f is $Q_t(s_t, u_t)$ for the FIS and f is u for the FLC, as follows

$$\frac{\partial Q_t(s_t, u_t)}{\partial K_l}=\overline{\omega}_l \tag{25}$$

Then we can obtain $\partial Q_t(s_t, u_t)/\partial\phi$ for the input parameters, σ_i^l and m_i^l, based on the chain rule,

$$\frac{\partial Q_t(s_t, u_t)}{\partial \sigma_i^l}=\frac{\partial Q_t(s_t, u_t)}{\partial \omega_l}\frac{\partial \omega_l}{\partial \sigma_i^l} \tag{26}$$

$$\frac{\partial Q_t(s_t, u_t)}{\partial m_i^l}=\frac{\partial Q_t(s_t, u_t)}{\partial \omega_l}\frac{\partial \omega_l}{\partial m_i^l} \tag{27}$$

The term $\partial Q_t(s_t, u_t)/\partial\omega_l$ is calculated from (14) and (12). The terms $\partial\omega_l/\partial\sigma_i^l$ and $\partial\omega_l/\partial m_i^l$ are calculated from both (11) and (10); hence

$$\frac{\partial Q_t(s_t, u_t)}{\partial \sigma_i^l}=\frac{(K_l-Q_t(s_t, u_t))}{\sum_l \omega_l}\omega_l\frac{2(x_i-m_i^l)^2}{(\sigma_i^l)^3} \tag{28}$$

$$\frac{\partial Q_t(s_t, u_t)}{\partial m_i^l}=\frac{(K_l-Q_t(s_t, u_t))}{\sum_l \omega_l}\omega_l\frac{2(x_i-m_i^l)}{(\sigma_i^l)^2} \tag{29}$$

Substituting from (25), (28) and (29) into (6) and modifying (24) to use eligibility trace, then the update law for the FIS parameters becomes

$$\phi_Q(t+1)=\phi_Q(t)+\eta\Delta_t e_t \tag{30}$$

The update law in (24) is applied also to the FLC by replacing $Q_t(s_t, u_t)$ with the output of the FLC, u. In addition and as shown from Figure 4, a random Gaussian noise, $n(0, \sigma_n)$, with zero mean and a standard deviation σ_n is added to the output of the FLC to solve the exploration/exploitation dilemma as for example the ε-greedy exploration method used in the discrete state–action space. Therefore, the update law for the FLC parameters is defined as

$$\phi_u(t+1)=\phi_u(t)+\xi\Delta_t \frac{\partial u}{\partial \phi}\left(\frac{u_c-u}{\sigma_n}\right) \tag{31}$$

where u_c is the output of the random Gaussian noise generator and ξ is the learning rate for the FLC parameters. The term $\partial u/\partial\phi$ can be calculated by replacing $Q_t(s_t, u_t)$ with the output of the FLC, u, in (25), (28), and (29).

DOI: 10.1002/acs

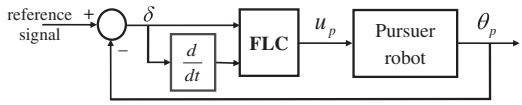

Figure 5. Block diagram of an FLC system.

5.1. Fuzzy logic controller

A block diagram of an FLC system is shown in Figure 5. The FLC has two inputs, the error in the pursuer angle, δ_{p}, defined in (18), and its derivative, $\dot{\delta}_{\mathrm{p}}$, and the output is the steering angle, u_{p}. We use the Gaussian MFs for the inputs of the FLC. We also modify (8) to be

$$R_l\colon \text{IF } \delta_{\mathrm{p}} \text{ is } A_1^l \text{ AND } \dot{\delta}_{\mathrm{p}} \text{ is } A_2^l \text{ THEN } f_l = K_l \tag{32}$$

where $l = 1, 2, \ldots, 9$. The crisp output, u_{p}, is calculated using (9) as follows:

$$u_{\mathrm{p}} = \frac{\sum_{l=1}^{9} \left(\prod_{i=1}^{2} \mu^{A_i^l}(x_i) \right) K_l}{\sum_{l=1}^{9} \left(\prod_{i=1}^{2} \mu^{A_i^l}(x_i) \right)} \tag{33}$$

5.2. Constructing the reward function

How to choose the reward function is very important in RL, because the agent depends on the reward function in updating its value function. The reward function differs from one system to another according to the desired task. In our case, the pursuer wants to catch the evader in minimum time. In other words, the pursuer wants to decrease its distance to the evader at each time step. The distance between the pursuer and the evader at time t is calculated as follows:

$$D(t) = \sqrt{(x_{\mathrm{e}}(t) - x_{\mathrm{p}}(t))^2 + (y_{\mathrm{e}}(t) - y_{\mathrm{p}}(t))^2} \tag{34}$$

The difference between two successive distances, $\Delta D(t)$, is calculated as

$$\Delta D(t) = D(t) - D(t+1) \tag{35}$$

A positive value of $\Delta D(t)$ means that the pursuer approaches the evader. The maximum value of $\Delta D(t)$ is defined as

$$\Delta D_{\max} = V_{\mathrm{rmax}} T \tag{36}$$

where V_{rmax} is the maximum relative velocity ($V_{\mathrm{rmax}} = V_{\mathrm{p}} + V_{\mathrm{e}}$) and T is the sampling time. Hence, we choose the reward, r, to be

$$r_{t+1} = \frac{\Delta D(t)}{\Delta D_{\max}} \tag{37}$$

6. COMPUTER SIMULATION

We use an INTEL CORE 2 DUO computer with a 2.0 GHz clock frequency and 4.0 GB of RAM. We do computer simulation with MATLAB software. Q(λ)-learning has many parameters to be set *a priori*; therefore, we tested computer simulation for different parameter values and different parameter value combinations and chose the values that give the best performance. The initial position of the evader is randomly chosen from a set of 64 different positions in the space.

We test the proposed technique in three cases. In case 1, the evader is constrained to use a simple control strategy (as the one used in [35]). The control strategy that is used by the evader is to run away from the pursuer along the LOS. In case 2, the evader uses a control strategy that makes

DOI: 10.1002/acs

use of its higher maneuverability (as the one used in [11, 46]). From the learning point of view, case 2 is the same as case 1 in which only the pursuer learns its control strategy. The reason for simulating case 2 is to show that even if the evader takes advantage of its higher maneuverability, the pursuer will still be able to learn to catch the evader.

Case 3 is different from case 1 and case 2 where we move on from learning in a single robot system to learning in a multi-robot system. In this case, both the pursuer and the evader learn their control strategies. In this case, the evader cannot learn to use the advantage of its higher maneuverability because the inputs to the FLC of the evader consists of the error in the angle of the LOS and its derivative and not the distance between the robots. As such the evader has no information on which to base a potential turning action. Therefore, the evader cannot learn how or when to turn. However, in future work, we will include position as an input to the FLC and then we hypothesize that the evader will then learn how and when to turn.

In our work, we want to show that our proposed technique (when it is compared with the others) is the closest technique to the classical control strategy. Given that the inputs to the evader are only the angle of the LOS and its rate of change, then the 'classical' strategy is also the 'optimal' strategy. Therefore, if we succeed in learning the 'classical' strategy, in this particular case, then the robots have also learned the 'optimal' strategy for this case.

6.1. The pursuit–evasion game

The pursuer starts motion from the position $(0, 0)$ with an initial orientation $\theta_p = 0$ and with a constant velocity $V_p = 2\,\text{m/s}$. The wheelbase $L_p = 0.3\,\text{m}$ and the steering angle $u_p \in [-0.5, 0.5]$. From (16), $Rd_{p_{min}} \simeq 0.55\,\text{m}$.

The evader starts motion from a random position for each episode with an initial orientation $\theta_e = 0$ and with a constant velocity $V_e = 1.0\,\text{m/s}$, which is half that of the pursuer (slower). The wheelbase $L_e = 0.3\,\text{m}$ and the steering angle $u_e \in [-1, 1]$, which is twice that of the pursuer (more maneuverable). From (16), $Rd_{e_{min}} \simeq 0.19\,\text{m}$ which is about one third that of the pursuer. The duration of a game is 60 s. The game ends when 60 s passed without capturing or when the capture occurs before the end of this time. The capture radius $\ell = 0.10\,\text{m}$. The sampling time is set to 0.1 s.

6.2. The proposed QLFIS

We choose the number of episodes (games) to be 1000, the number of plays (steps) in each episode to be 600, $\gamma = 0.95$, and $\lambda = 0.9$. We make the learning rate for the FIS, η, decrease with each episode such that

$$\eta = 0.1 - 0.09\left(\frac{i}{\text{Max. Episodes}}\right) \tag{38}$$

and also make the learning rate for the FLC, ξ, decrease with each episode such that

$$\xi = 0.01 - 0.009\left(\frac{i}{\text{Max. Episodes}}\right) \tag{39}$$

where i is the current episode. Note that the value of η is 10 times the value of ξ i.e. the FIS converges faster than the FLC to avoid instability in tuning the parameters of the FLC. We choose $\sigma_n = 0.08$.

6.3. Compared techniques

To validate the proposed QLFIS technique, we compare its results with the results of the classical control strategy, Q(λ)-learning only and the technique proposed in [19]. The classical control strategies of the pursuer and the evader are defined by (17) and (18). The parameters of Q(λ)-learning only have the following values: the number of episodes is set to 1000, the number of

DOI: 10.1002/acs

plays in each episode to 6000, $\gamma = 0.5$, and $\lambda = 0.3$. We make the learning rate, α, decrease with each episode such that

$$\alpha = \left(\frac{1}{i}\right)^{0.7} \tag{40}$$

We also make ε decrease with each episode such that

$$\varepsilon = \frac{0.1}{i} \tag{41}$$

where i is the current episode.

For the technique proposed in [19], we choose the same values for the parameters of the NN. The NN has a three-layer structure with 7-21-1 nodes. The RL parameters and the initial values of the input and the output parameters of the FLC are all chosen to be the same as those chosen in the proposed QLFIS. We choose $\sigma_n = 0.1$, which decreases each episode by $1/i$ where i is the current episode.

7. RESULTS

Figure 6 and Table I show the input and the output parameters of the FLC after tuning using the proposed QLFIS, respectively where '**N**', '**Z**', and '**P**' are referred to the linguistic values '**N**egative', '**Z**ero', and '**P**ositive'.

Table II shows the capture times for different initial positions of the evader using the classical control strategy of the pursuer, the Q(λ)-learning only, the proposed QLFIS and the technique proposed in [19]. In addition, the learning times for the different techniques are also shown in this table. From Table II we can see that although the Q(λ)-learning only has the minimum learning time, it is not enough to get the desired performance in comparison with the classical control strategy and the other techniques. The proposed QLFIS outperforms the technique proposed in

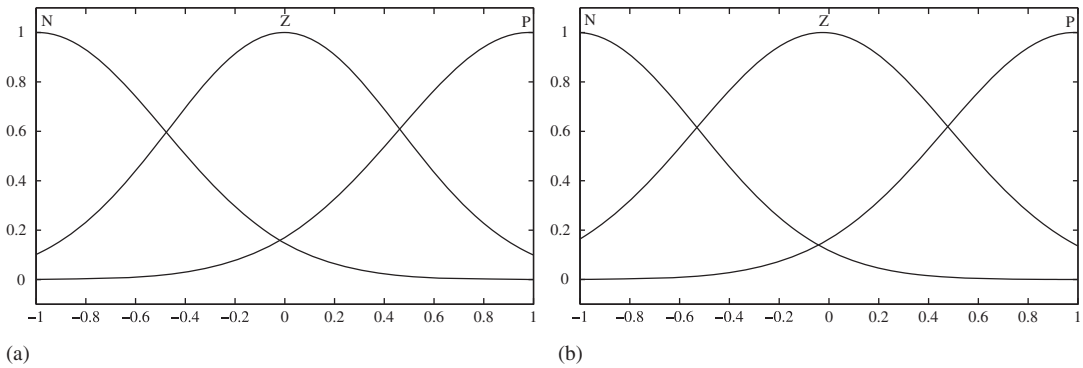

Figure 6. MFs for the inputs after tuning using the proposed QLFIS: (a) the input δ_p and (b) the input $\dot{\delta}_p$.

Table I. Fuzzy decision table after tuning using the proposed QLFIS.

		$\dot{\delta}_p$		
		N	*Z*	*P*
δ_p	*N*	−0.5452	−0.2595	−0.0693
	Z	−0.2459	0.0600	0.2299
	P	0.0235	0.3019	0.5594

DOI: 10.1002/acs

Table II. Capture time, in seconds, for different evader initial positions and learning time, in seconds, for the different techniques (case 1).

	Evader initial position				Learning time
	$(-6,7)$	$(-7,-7)$	(2,4)	$(3,-8)$	
Classical control strategy	9.6	10.4	4.5	8.5	—
Q(λ)-learning only	12.6	15.6	8.5	11.9	32.0
Technique proposed in [19]	10.9	12.9	4.7	9.1	258.6
Proposed QLFIS	10.0	10.7	4.6	8.8	65.2

[19] in both performance and learning time and both of them have better performance than using the Q(λ)-learning only. We can also see that the proposed QLFIS takes only 65.2 s in the learning process, which is about 25% of the learning time taken by the technique proposed in [19].

Note that in the previous game, the evader uses a simple control strategy. Now we will use another version of the pursuit–evasion game in which we will make use of the advantage of the maneuverability of the evader during the game. Equations of motion for the pursuer and the evader are modified to be [11, 46]

$$\begin{aligned} \dot{x}_i &= v_i \cos(\theta_i) \\ \dot{y}_i &= v_i \sin(\theta_i) \\ \dot{\theta}_i &= \frac{v_i}{L_i} \tan(u_i) \end{aligned} \tag{42}$$

where v_i is the velocity of the robot which is governed by the steering angle, to avoid slips, such that

$$v_i = V_i \cos(u_i) \tag{43}$$

where V_i is the maximum velocity. We will also modify the strategy of the evader to make use of its higher maneuverability during the game as follows:

1. If the distance between the pursuer and the evader is greater than a certain amount, d, then the control strategy for the evader is

$$u_{\mathrm{e}} = \tan^{-1}\left(\frac{y_{\mathrm{e}} - y_{\mathrm{p}}}{x_{\mathrm{e}} - x_{\mathrm{p}}}\right) - \theta_{\mathrm{e}} \tag{44}$$

2. If the distance between the pursuer and the evader is smaller than d, then the control strategy for the evader is

$$u_{\mathrm{e}} = (\theta_{\mathrm{p}} + \pi) - \theta_{\mathrm{e}} \tag{45}$$

The distance d defined in this strategy is calculated as

$$d = \frac{L_{\mathrm{p}}}{\tan(u_{\mathrm{p_{max}}})} \tag{46}$$

where d is the same as the minimum turning radius of the pursuer, $Rd_{\mathrm{p_{min}}}$, defined by (16). Equations (45) and (46) mean that if the pursuer approaches the evader and the distance between them is equal to the minimum turning radius of the pursuer, then the evader makes use of its higher maneuverability by turning to take the opposite direction of the pursuer.

Figure 7 and Table III show the input and the output parameters of the FLC after tuning using the proposed QLFIS, respectively. Table IV shows the capture times for different initial positions of the evader using the classical control strategy of the pursuer, the Q(λ)-learning only, the proposed QLFIS, and the technique proposed in [19]. In addition, the learning times for the different techniques are also shown in this table. The results shown in Table IV agree with those

DOI: 10.1002/acs

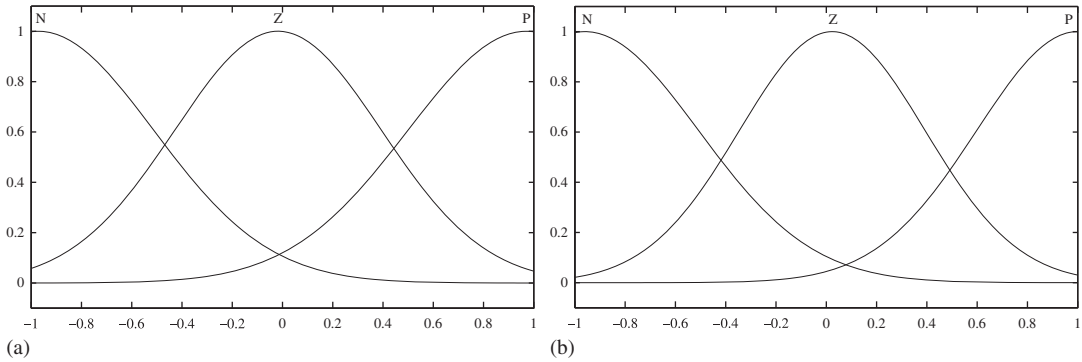

Figure 7. MFs for the inputs after tuning using the proposed QLFIS: (a) the input δ_p and (b) the input $\dot{\delta}_p$.

Table III. Fuzzy decision table after tuning using the proposed QLFIS.

		$\dot{\delta}_p$		
		N	*Z*	*P*
δ_p	*N*	−1.2211	−0.4285	−0.6713
	Z	0.0014	−0.0001	0.0422
	P	0.5187	0.4481	1.3302

Table IV. Capture time, in seconds, for different evader initial positions and learning time, in seconds, for the different techniques (case 2).

	Evader initial position				
	(−7, 5)	(−4, −6)	(3,5)	(5, −4)	Learning time
Classical control strategy	10.3	9.1	7.3	8.0	—
Q(λ)-learning only	15.6	14.3	12.4	12.9	76.7
Technique proposed in [19]	12.4	11.3	9.1	9.8	323.2
Proposed QLFIS	10.6	9.3	7.4	8.1	136.6

shown in Table II. Figure 8 shows the paths of the pursuer and the evader when the proposed QLFIS is used. From this figure we can see that the evader can first escape from the pursuer by turning quickly, when the distance between them is equal to the minimum turning radius of the pursuer. However, the pursuer can modify his path using its learned control strategy and can successfully catch the evader.

Now, we will increase the complexity of the model by making both the pursuer and the evader self-learn their control strategies simultaneously. The complexity in the learning process is that each robot will try to find its control strategy based on the control strategy of the other robot which, at the same time, is still learning.

Figure 9 and Table V show the input and the output parameters of the tuned FLC for the pursuer using the proposed QLFIS. Figure 10 and Table VI show the input and the output parameters of the tuned FLC for the evader using the proposed QLFIS.

To check the performance of the different techniques, we cannot use the capture time as a measure, as we did in Table II; because in this game both the pursuer and the evader are learning, hence we may find capture times that are smaller than those corresponding to the classical solution. Of course that does not mean that the performance is better than the classical solution, but it means that the evader does not learn well and as a result it is captured in a shorter time. Therefore, the

DOI: 10.1002/acs

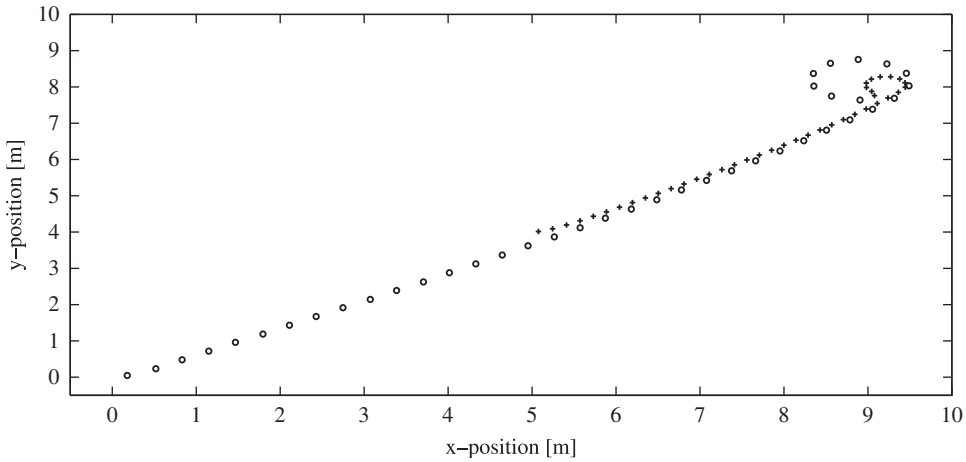

Figure 8. Paths of the pursuer (ooo) and the evader (+++) using the proposed QLFIS.

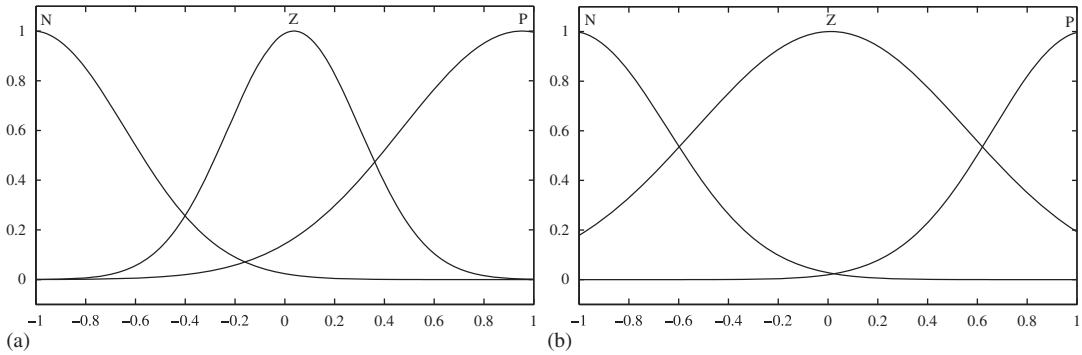

Figure 9. MFs for the inputs of the pursuer after tuning using the proposed QLFIS: (a) the input δ_p and (b) the input $\dot{\delta}_p$.

Table V. Fuzzy decision table for the pursuer after tuning using the proposed QLFIS.

		$\dot{\delta}_p$		
		N	*Z*	*P*
δ_p	*N*	−1.2990	−0.6134	−0.4064
	Z	−0.3726	−0.0097	0.3147
	P	0.3223	0.5763	0.9906

measure that we use is the paths of both the pursuer and the evader instead of the capture time. Figures 11–13 show the paths of the pursuer and the evader of the different techniques against the classical control strategies of the pursuer and the evader.

We can see that the best performance is that of the proposed QLFIS. Table VII shows the learning time for the different techniques. From this table we can see that the proposed QLFIS still has the minimum learning time when compared with the technique proposed in [19]. Finally, we can conclude that the proposed QLFIS still outperforms the technique proposed in [19] in both performance and learning time.

DOI: 10.1002/acs

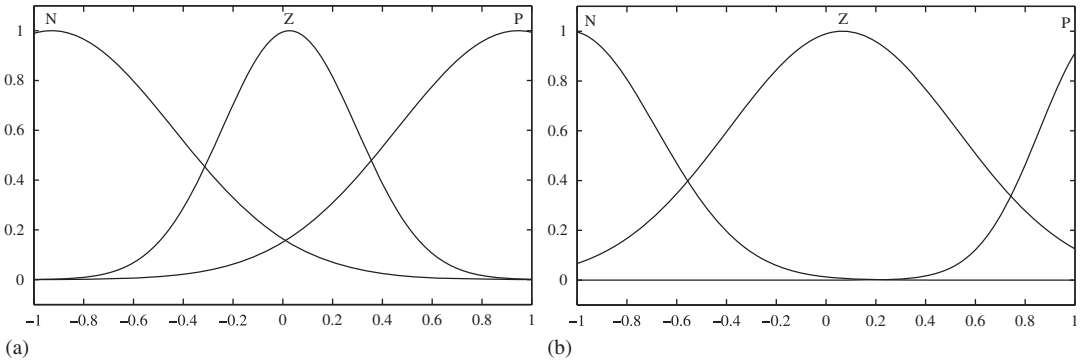

Figure 10. MFs for the inputs of the evader after tuning using the proposed QLFIS: (a) the input δ_e and (b) the input $\dot{\delta}_e$.

Table VI. Fuzzy decision table for the evader after tuning using the proposed QLFIS.

		$\dot{\delta}_e$		
		N	*Z*	*P*
δ_e	*N*	−1.4827	−0.4760	−0.0184
	Z	−0.5365	−0.0373	0.5500
	P	−0.0084	0.4747	1.1182

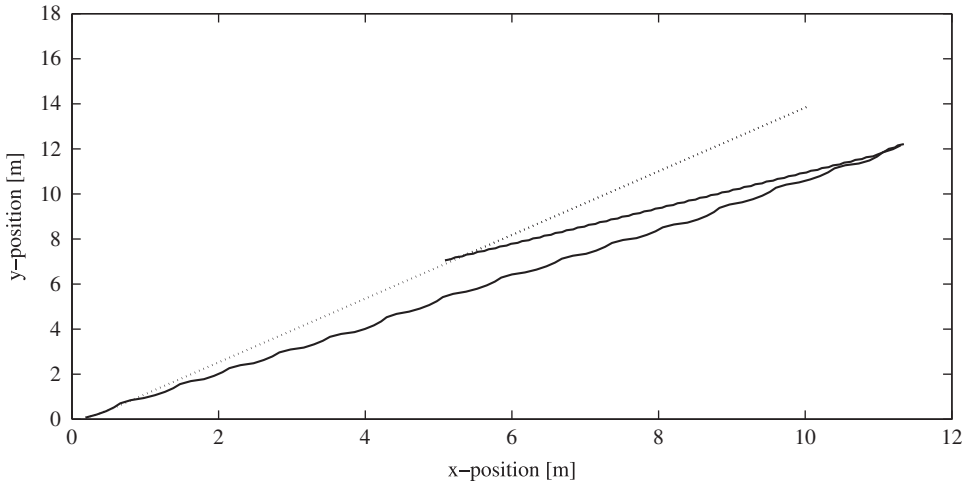

Figure 11. Paths of the pursuer and the evader using the Q(λ)-learning only (solid line) against the classical strategies (dotted line).

8. CONCLUSION

In this paper, we proposed a novel technique to tune the input and the output parameters of an FLC in which RL is combined with FIS as a function approximation to generalize the state and the action spaces to the continuous case. The proposed technique is applied to three different pursuit–evasion games. We start with a simple game in which we assume that the pursuer does not know its control strategy while the evader plays a simple classical control strategy. However, the pursuer can self-learn its control strategy by interaction with the evader. Then, we make use of the advantage of maneuverability of the evader during the game. Finally, we apply the proposed

DOI: 10.1002/acs

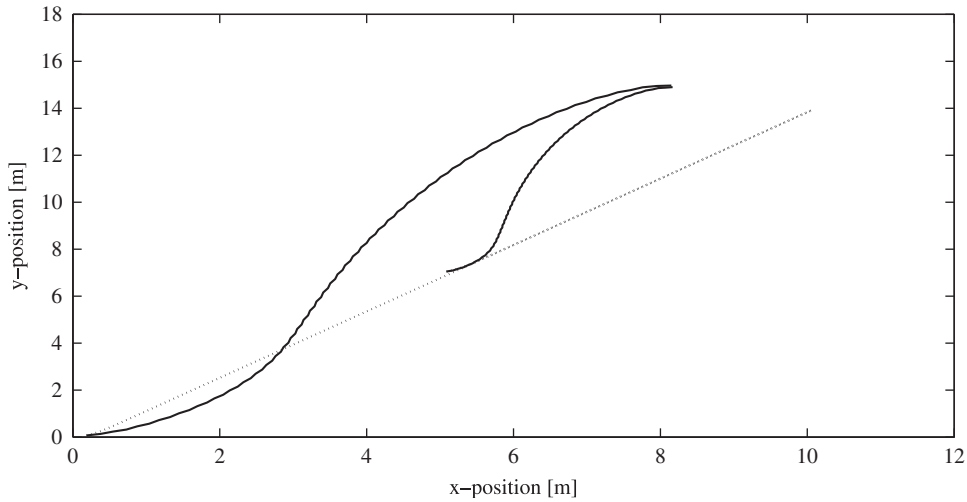

Figure 12. Paths of the pursuer and the evader using the technique proposed in [19] (solid line) against the classical strategies (dotted line).

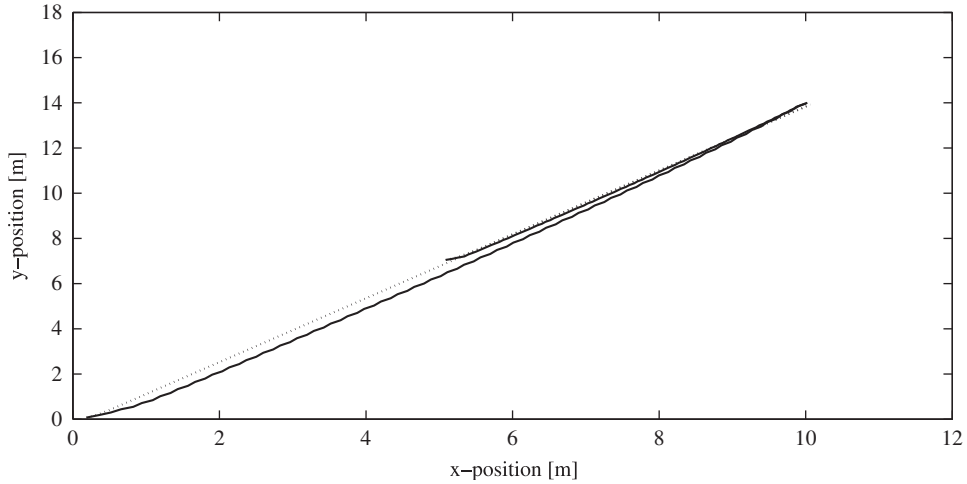

Figure 13. Paths of the pursuer and the evader using the proposed QLFIS (solid line) against the classical strategies (dotted line).

Table VII. Learning time, in seconds, for the different techniques.

	Learning time
Q(λ)-learning only	62.8
Technique proposed in [19]	137.0
Proposed QLFIS	110.3

technique to a pursuit–evasion game in which both the pursuer and the evader do not know their control strategies. In this case, the evader has no information on which to base a potential turning action. Therefore, the evader cannot learn how or when to turn. However, in future work, we will include position as an input to the FLC and then we hypothesize that the evader will then learn how and when to turn. Computer simulation and the results show that the proposed QLFIS technique outperforms the other techniques in performance when compared with the classical control strategies and in the learning time, which is also an important factor especially in online applications.

DOI: 10.1002/acs

REFERENCES

1. Micera S, Sabatini AM, Dario P. Adaptive fuzzy control of electrically stimulated muscles for arm movements. *Medical and Biological Engineering and Computing* 1999; **37**(6):680–685.
2. Daldaban F, Ustkoyuncu N, Guney K. Phase inductance estimation for switched reluctance motor using adaptive neuro-fuzzy inference system. *Energy Conversion and Management* 2006; **47**:485–493.
3. Labiod S, Guerra TM. Adaptive fuzzy control of a class of SISO nonaffine nonlinear systems. *Fuzzy Sets and Systems* 2007; **158**(10):1126–1137.
4. Lam HK, Leung FHF. Fuzzy controller with stability and performance rules for nonlinear systems. *Fuzzy Sets and Systems* 2007; **158**:147–163.
5. Hagras H, Callaghan V, Colley M. Learning and adaptation of an intelligent mobile robot navigator operating in unstructured environment based on a novel online fuzzy-genetic system. *Fuzzy Sets and Systems* 2004; **141**(1):107–160.
6. Mucientes M, Moreno DL, Bugarin A, Barro S. Design of a fuzzy controller in mobile robotics using genetic algorithms. *Applied Soft Computing* 2007; **7**(2):540–546.
7. Wang LX. Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems*, *Man*, *and Cybernetics* 1992; **22**(6):1414–1427.
8. Herrera F, Lozano M, Verdegay JL. Tuning fuzzy logic controllers by genetic algorithms. *International Journal of Approximate Reasoning* 1995; **12**:299–315.
9. Lekova A, Mikhailov L, Boyadjiev D, Nabout A. Redundant fuzzy rules exclusion by genetic algorithms. *Fuzzy Sets and Systems* 1998; **100**:235–243.
10. Desouky SF, Schwartz HM. Genetic based fuzzy logic controller for a wall-following mobile robot. *2009 American Control Conference* (*ACC-09*), St. Louis, MO. IEEE: New York, 2009; 3555–3560.
11. Desouky SF, Schwartz HM. Different hybrid intelligent systems applied for the pursuit–evasion game. *2009 IEEE International Conference on Systems*, *Man*, *and Cybernetics*, San Antonio, TX, 2009; 2677–2682.
12. Sutton RS, Barto AG. *Reinforcement Learning*: *An Introduction*. MIT Press: Cambridge, MA, 1998.
13. Kaelbling LP, Littman ML, Moore AP. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research* 1996; **4**:237–285.
14. Smart WD, Kaelbling LP. Effective reinforcement learning for mobile robots. *IEEE International Conference on Robotics and Automation*, vol. 4, Washington, D.C., 2002; 3404–3410.
15. Ye C, Yung NHC, Wang D. A fuzzy controller with supervised learning assisted reinforcement learning algorithm for obstacle avoidance. *IEEE Transactions on Systems*, *Man and Cybernetics*, *Part B*: *Cybernetics* 2003; **33**(1): 17–27.
16. Kondo T, Ito K. A reinforcement learning with evolutionary state recruitment strategy for autonomous mobile robots control. *Robotics and Autonomous Systems* 2004; **46**(2):111–124.
17. Gutnisky DA, Zanutto BS. Learning obstacle avoidance with an operant behavior model. *Artificial Life* 2004; **10**(1):65–81.
18. Rodríguez M, Iglesias R, Regueiro CV, Correa J, Barro S. Autonomous and fast robot learning through motivation. *Robotics and Autonomous Systems* 2007; **55**(9):735–740.
19. Dai X, Li CK, Rad AB. An approach to tune fuzzy controllers based on reinforcement learning for autonomous vehicle control. *IEEE Transactions on Intelligent Transportation Systems* 2005; **6**(3):285–293.
20. Er MJ, Deng C. Obstacle avoidance of a mobile robot using hybrid learning approach. *IEEE Transactions on Industrial Electronics* 2005; **52**(3):898–905.
21. Xiao H, Liao L, Zhou F. Mobile robot path planning based on Q-ANN. *IEEE International Conference on Automation and Logistics*, Jinan, China, 2007; 2650–2654.
22. Vidal R, Shakernia O, Kim HJ, Shim DH, Sastry S. Probabilistic pursuit–evasion games: theory, implementation, and experimental evaluation. *IEEE Transactions on Robotics and Automation* 2002; **18**(5):662–669.
23. Li D, Chen G, Kwan C, Chang M. A hierarchical approach to multi-player pursuit–evasion differential games. *Forty-forth IEEE Conference on Decision and Control*, *and the European Control Conference 2005*, Seville, Spain, 2005; 5674–5679.
24. Ishiwaka Y, Satob T, Kakazu Y. An approach to the pursuit problem on a heterogeneous multiagent system using reinforcement learning. *Robotics and Autonomous Systems* 2003; **43**(4):245–256.
25. Givigi SN, Schwartz HM, Lu X. A reinforcement learning adaptive fuzzy controller for differential games. *Journal of Intelligent and Robotic Systems* 2010; **59**(1):3–30.
26. Jouffe L. Fuzzy inference system learning by reinforcement methods. *IEEE Transactions on Systems*, *Man*, *and Cybernetics*, *Part C*: *Applications and Reviews* 1998; **28**(3):338–355.
27. Duan Y, Hexu X. Fuzzy reinforcement learning and its application in robot navigation. *International Conference on Machine Learning and Cybernetics*, Guangzhou, vol. 2. IEEE: New York, 2005; 899–904.
28. Waldock A, Carse B. Fuzzy q-learning with an adaptive representation. *2008 IEEE 16th International Conference on Fuzzy Systems* (*FUZZ-IEEE*), Piscataway, NJ, 2008; 720–725.
29. Watkins CJCH, Dayan P.Qlearning. *Machine Learning* 1992; **8**(3):279–292.
30. Watkins CJCH. Learning from delayed rewards. *Ph.D. Thesis*, Cambridge University, 1989.
31. Bhanu B, Leang P, Cowden C, Lin Y, Patterson M. Real-time robot learning. *IEEE International Conference on Robotics and Automation*, Seoul, Korea, vol. 1, 2001; 491–498.

DOI: 10.1002/acs

32. Maček K, Petrović I, Perić N. A reinforcement learning approach to obstacle avoidance of mobile robots. *Seventh International Workshop on Advanced Motion Control*, Maribor, Slovenia, 2002; 462–466.
33. Marin TM, Duckett T. Fast reinforcement learning for vision-guided mobile robots. *IEEE International Conference on Robotics and Automation*, Barcelona, Spain, 2005; 4170–4175.
34. Yang Y, Tian Y, Mei H. Cooperative Q-learning based on blackboard architecture. *International Conference on Computational Intelligence and Security Workshops*, Harbin, China, 2007; 224–227.
35. Desouky SF, Schwartz HM. A novel technique to design a fuzzy logic controller using Q(λ)-learning and genetic algorithms in the pursuit–evasion game. *2009 IEEE International Conference on Systems*, *Man*, *and Cybernetics*, San Antonio, TX, 2009; 2683–2689.
36. Desouky SF, Schwartz HM. A novel hybrid learning technique applied to a self-learning multi-robot system. *2009 IEEE International Conference on Systems*, *Man*, *and Cybernetics*, San Antonio, TX, 2009; 2690–2697.
37. Stephan V, Debes K, Gross HM. A new control scheme for combustion processes using reinforcement learning based on neural networks. *International Journal of Computational Intelligence and Applications* 2001; **1**(2): 121–136.
38. Yan XW, Deng ZD, Sun ZQ. Genetic Takagi–Sugeno fuzzy reinforcement learning. *IEEE International Symposium on Intelligent Control*, Mexico City, Mexico, 2001; 67–72.
39. Gu D, Hu H. Accuracy based fuzzy Q-learning for robot behaviours. *International Conference on Fuzzy Systems*, Budapest, Hungary, vol. 3, 2004; 1455–1460.
40. Takagi T, Sugeno M. Fuzzy identification of systems and its applications to modelling and control. *IEEE Transactions on Systems*, *Man and Cybernetics* 1985; **SMC-15**(1):116–132.
41. Isaacs R. *Differential Games*. Wiley: New York, 1965.
42. LaValle SM. *Planning Algorithms*. Cambridge University Press: Cambridge, 2006.
43. Peng J, Williams RJ. Incremental multi-step Q-learning. *Machine Learning* 1996; **22**(1–3):283–290.
44. Jouffe L. Actor-critic learning based on fuzzy inference system. *IEEE International Conference on Systems*, *Man*, *and Cybernetics*, Beijing, China, vol. 1, 1996; 339–344.
45. Wang XS, Cheng YH, Yi JQ. A fuzzy actor-critic reinforcement learning network. *Information Sciences* 2007; **177**:3764–3781.
46. Lim SH, Furukawa T, Dissanayake G, Whyte HFD. A time-optimal control strategy for pursuit–evasion games problems. *International Conference on Robotics and Automation*, New Orleans, LA, 2004.

DOI: 10.1002/acs