International Journal of Physical Sciences Vol. 7(43), pp. 5736-5753, 16 November, 2012
Available online at http://www.academicjournals.org/IJPS
DOI: 10.5897/IJPS11.1477
ISSN 1992 - 1950 ©2012 Academic Journals

Full Length Research Paper

Experimental investigations of immune fuzzy Q-learning algorithms for robot obstacle avoidance

Suchart Punpaisarn and Sarawut Sujitjorn*

The School of Electrical Engineering, Suranaree University of Technology, Nakhon Ratchasima, 30000, Thailand.

Accepted 17 February, 2012

This article presents a new approach to car-like robot control for obstacle avoidance and target tracking. The proposed approach employs cooperative algorithms including artificial immune algorithms, fuzzy logic and Q-learning denoted shortly as IFQ-learning control. The article explains the artificial immune system and the proposed algorithms. The fuzzy Q-learning algorithms are also presented. The article elaborates the control design as well as extensive experimental results. Very satisfactory robot performances are achieved via the proposed IFQ-learning control. VDO clips illustrating the experiments are available on the website http://www.sut.ac.th/engineering/electrical/carg/.

Key words: Q-learning, artificial immune, fuzzy logic, robot, obstacle avoidance.

INTRODUCTION

In recent years, researchers on autonomous mobile robot systems have required controllers capable of solving navigation problems in uncertain environmental situations. Fuzzy logic control has been well received for such purposes since it has an ability to use expert knowledge expressed in the form of linguistic rules. Inference under an uncertain environment is also possible. To appropriately design a fuzzy rule-base and membership functions for a control system is tedious and quite often requires trial-and-error testing. The manually tuned fuzzy rules are not optimal, and they also cause an increasing number of variables in a fuzzy control system. Parameter tuning of membership functions is a time consuming task. Xu and Tso (1999) proposed a reactive behavior-based fuzzy logic controller with a virtual target technique. It tries to utilize the rules to define the robot reaction to unknown environments, and applies fuzzy inference to coordinate different reactive behaviors. However, the dead cycle problem (going around in circles or cycling between multiple traps) occurs in some cases. Er and Deng (2004) proposed dynamic fuzzy Q-learning for reinforcement fuzzy control for a mobile robot wall-following. Although their fuzzy controller is active in an online manner, some prior knowledge is introduced beforehand through an offline process.

To improve the performance of a fuzzy controller, there are several techniques to develop and achieve their tunings of successive parameters. A hybrid intelligent controller is a common solution. Senthilkumar and Bharadwaj (2009) applied the hybrid genetic-fuzzy reinforcement learning in an autonomous mobile robot system. Their objective is to improve fuzzy rules governing the actions and behaviors of navigation and obstacle avoidance. Genes are represented in the form of distances and angles labels. The chromosomes are represented as a rule written in a Boolean algebraic form. Hagras et al. (2000) developed an online-learning genetic algorithm for learning the membership functions of behaviors of an autonomous mobile robot. The fuzzy controller is used to reduce the number of rules, while the genetic algorithm is applied to tune the membership functions. Juang and Lu (2009) presented the design of fuzzy controllers by ant colony optimization (ACO) incorporated with fuzzy-Q learning. Their simulation results show good performances in a small scale of pheromone trail limitation settings. In Juang and Hsu (2009), a reinforcement ant optimized fuzzy controller (RAOFC) applied to a wheeled mobile robot wall following control is described. The heuristic of Q-value is used to improve learning for ant colony optimization. Experiment of RAOFC with the PIONEER 3-DX mobile robot illustrates the effectiveness and efficiency of the robot

*Corresponding author. E-mail: sarawut@sut.ac.th. Tel: (+66) 4422-4402. Fax: (+66) 4422-4601.

moving along a wall. Amin and Adriansyah (2006) proposed fuzzy membership functions and fuzzy rule-base tunings using particle swarm fuzzy controller (PSFC). With this approach, the obtained fuzzy controller can govern the mobile robot to have smooth movements in simulated environments. Learning capability of a fuzzy controller has been achieved for a number of years by using neural networks (NNs). The NNs help optimize the fuzzy rules and the fuzzy memberships. As a result, robots with learning fuzzy controllers have better performances in terms of smoothness, and obstacle avoidance (Rusu et al., 2003; Zhu and Yang, 2007; Parhi and Sing, 2009; Garbi et al., 2007). Since several means are available to achieve robot learning control, this paper proposes a novel cooperative approach of artificial immune algorithm, fuzzy logic and Q-learning for autonomous mobile robot obstacle avoidance in unknown environments. In other words, the fuzzy-Q learning controller is enhanced by the artificial immune algorithm. Our investigation compares the fuzzy control, the fuzzy-Q learning control with the proposed method. Experimental results confirm that the proposed immune based fuzzy-Q learning control provides superior results to the other approaches.

The introductory part of this article is followed by an explanation of the artificial immune system (AIS), the immune algorithm, the Q-learning, the fuzzy-Q learning (FQ) and the proposed immune fuzzy-Q learning (IFQ). Next is an explanation of the structure and hardware components of the car-like robot used, followed by an elaboration of the control design. Experimental results and discussions covering the comparisons between the simple fuzzy logic, the FQ and the proposed IFQ controllers can be found in this study. VDO clips are available on website *http://www.sut.ac.th/engineering/electrical/carg/*; and then conclusions.

LEARNING CONTROL

Learning has been an exciting research area in control for many years. Yet, a clear definition of learning control is still debatable. Albus (1991), Antsaklis (1994), and Farrell and Baker (1995) agree with the definition of learning as gaining mastery through experience and fixing in mind or memory, then acquiring experience, ability or skill. Advantageously, learning control can lead to an autonomous dynamical system. Learning control can be classified as *learning about the plant*, *learning about the environment*, *learning about the controller*; and *learning new goals and constraints* (Antaklis, 1994). The implementation of a learning system requires *performance feedback, memory, and training.* Hence, learning system should be capable of evaluating its current and past performances, memorizing useful knowledge for a future use, and translating the quantitative information about performance into a memorized form of knowledge. It can be said that learning is an estimation or successive approximation of unknown operations.

Artificial immune system (AIS)

The biological immune system is a complex adaptive system. In medical science, immunity refers to the condition in which an organism can resist disease, more specifically infectious disease. Cells and molecules constitute the biological immune system, which can be imagined as a multilayer protection system containing different types of defense mechanism for detection, recognition and action. There exist three main layers in an immune system namely anatomic barrier, innate immunity and adaptive immunity, respectively. The innate and adaptive immunities are inter-linked and influence each other (Abbas and Lichtman, 2000). The innate immunity is an unchanging mechanism that detects and destroys certain invading organisms. The most important cells for protecting the body from infections are white blood cells (or T-cells), and large lymphocytes (anti-product cells or B-cells). There are three types of the T-cells: the T helper cells essential to the activation of the B-cells, the killer T-cells seizing and destroying invaders, and the suppressor T-cells, preventing allergic reactions and autoimmune diseases. The B-cells are co-responsible for the production and ejection of antibodies. In fact, they are specific proteins that catch the antigens and destroy them. The process can be represented by the diagram in Figure 1 proposed by de Castro and Zuben (2002), in which readers can find detailed explanation of medical immune system constituting of seven sub-processes shown by the diagram.

A novel computational intelligence technique called artificial immune system (AIS) is inspired by the biological immune system under the framework of immune network theory (Jerne, 1974). Castro gives the definition of AIS in Castro and Timmis (2002) as *"adaptive systems, inspired by theoretical immunology and observed immune functions, principles and models, which are applied to problem solving"*. Based on the knowledge of the human immune system, immune algorithms for computational purposes have been developed. The algorithms regard the problem to be solved as antigen (*Ag*), and the best solution vector as antibody (*Ab*). Lymphocyte differentiation is regarded as the maintenance of elite solutions in the memory cells. The T-cell suppression is the elimination of surplus candidate solutions. An immune system detector (lymphocyte or B-cells) has a high affinity (high level of matching in computation) with an antigen (invader microorganism). When an antibody strongly matches an antigen, the corresponding B-cell is stimulated to produce its clones then produce more antibodies. This process is called the clonal enlargement since new cells are produced by cloning and mutating

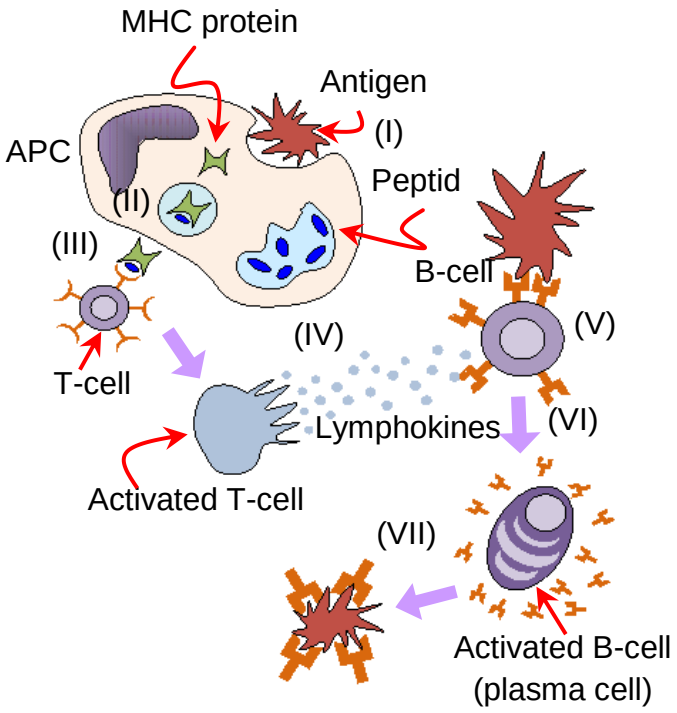

Figure 1. The acquired immune system mechanism.

existing cells. This clonal enlargement reaches to destroy or neutralize the antigen. The retention of some cells is memorized so that the immune system can act more quickly in future cases of the same antigens. An actual immune system has a specific sub-process called clonal selection that defines the reaction of system to antigens' invasion. Computationally, it is represented by sub-algorithm referred to as the clonal selection classifier. Li et al. (2009) presented a hybrid system namely an improved clonal selection classifier (ICSC) (Zhuang et al., 2009). Their algorithms combine the clonal selection classifier with the fuzzy C-mean clustering (FCM) for data classification. This improvement decreases the time consumed by the data classification process. The multi-layered immune inspired machine learning algorithm is also developed by Knight and Timmi (2003). They model many AIS algorithm units, and form an unsupervised learning system successfully used in classification applications.

This research presents an application of the AIS by classification task of fuzzy rules for mobile robot navigation. AIS develops a population of antibodies each of which represents the precedence ("IF part") of a fuzzy rule. Each antigen represents an experiment (record, or example case). The rule precedence is structured by a conjunction of conditions (attribute-value pairs). Each attribute can be structured either continuously or categorically. Categorical attributes are designed by expert, and continuous attributes are fuzzified by using a set of linguistic terms.

The immune algorithm presented below follows the clonal selection classifier in Castro and Timmis (2002). The main idea of this algorithm is the classification of information which is involved in the training set. This information has not been properly used so that it could be employed to produce classification performance enhancements. The list of the algorithm is as follows:

Immune algorithm

(1) Initialization: The information data are normalized to ensure that the Euclidean distance range of information is in the interval [0, 1]. The distance between *Ags* and *Abs* can be calculated using Equation 1

$$D = \sqrt{\sum_{k=1}^{N} (Ab_{i,k} - Ag_{i,k})^2} \quad (1)$$

Here $Ab_{i,k}$ and $Ag_{i,k}$ are the k^{th} attributes of the Ab_i and Ag_i, respectively, $i = 1,\ldots, Mc$, where Mc is the number of memory *Abs* for related class.

(2) Create memory pool (M) in matrix form.

(3) Antigenic presentation: For each antigenic pattern Ag_i from the antigen population Ag do $Ag_i = \{f_1,\ldots, f_n\}, i = 1,\ldots, N(Ag_i \in Ag)$; f_i is the input feature.

(3.1) Based-on a uniform distribution, randomly assign centers of clustering to initial antibodies, *Ab*.

(3.2) Calculate affinity vector according to f_i for all antibodies using the Euclidean distance between the vector Ag_i and the center *Ab.*
(3.2.1) Select *n* highest affinity antibodies from *Abs* to compose a set *Abn*.
(3.2.2) If the average affinity of the *Abn* is greater than a threshold value of 0.5 (Castro and Timmis, 2002) then go to (3.4).
(3.3) Proportionally proliferate *n* antibodies in *Abn* to their antigenic affinities.
(3.3.1) Based-on a uniform distribution, randomly generate a set *C* of clones.
(3.3.2) Generate a mutated antigen Ag_i in set *C**.
(3.3.3) Select *C** to compose the set *Ab*, and return to (3.2).
(3.4) Select the best affinity memory cell (*Mc*) of the same class for Ag_i, and the highest affinity antibody from *Abn* as a candidate memory (Mc_{cand}). (Remarks: The best affinity possesses the shortest distance between Ag_i and *Abn*. The distance is calculated using Equation 1.
(3.5) If the affinity of Mc_{cand} for Ag_i is better than that of the *Mc*, then add Mc_{cand} to the memory pool *M*.
(3.6) If the affinity between Mc_{cand} and *Mc* is below the affinity threshold (*AT*) then remove *Mc* from *M.* (*AT* can be set by Equation 2).

$$AT = \delta \times a \tag{2}$$

where:

a = A constant real value between 0 to 1,
δ = The average distance between each Ag_i in *Ag*s.
Regarding this, a = 0.8 is used as suggested by Castro and Timmis (2002).
(4) Classify information by performing in a K-nearest neighbor (KNN) approach from training set of memory pool *M*.

Q-learning

This research proposes reinforcement learning based on the Q-learning technique for fuzzy inference system. The technique works by learning an action value function (Wakins and Dayan, 1992). It gives the expected utility of a given action in state form. Q-learning can be embedded into the fuzzy rules to reduce training (Jouffe, 1998). The Q-function is used to estimate the future performing actions, and select a proper action for a particular state. The present state, $\vec{x}(t)$, of a robot is executed by an action *a(t)* according to the evaluation of the feedback value. The Q-function is used to evaluate and estimate the discounted cumulative reinforcement for taking actions from given states. The Q-function is a mapping technique from the pairs of the state-action to predict feedback value. Its output for state vector $\vec{x}$ and action (*a*) is signified by the Q-value ($Q(\vec{x}, a)$).

$$\begin{aligned} Q\{\vec{x}(t), a(t)\} \leftarrow Q^a\left(\vec{x}(t), a(t)\right) + \alpha \cdot [r(t+1) \\ + \gamma Q^b\left(\vec{x}(t+1)\right) \\ - Q^a\left(\vec{x}(t), a(t)\right)] \end{aligned} \tag{3}$$

Equation 3 represents the updating expression of the Q-function. Let $\vec{x}(t+1)$ be the obtained state from $\vec{x}(t)$ after the current action of *a(t)*, $r(t+1)$ is the scalar reinforcement signal that depends on environment, α is the learning rate, γ is the discount factor, *t* is the t^{th} iteration, and $Q^b(\vec{x}(t+1))$ is the best estimated Q-value that the agent assumes to reach the state $\vec{x}(t+1)$. The best estimation $Q^b(\vec{x}(t+1))$ can be defined by

$$Q^b(\vec{x}(t+1)) = \max_{b \in P(\vec{x}(t+1))} \{Q(\vec{x}(t+1), b)\} \tag{4}$$

where $P(\vec{x}(t+1))$ is the possible action sets at the state $\vec{x}(t+1)$. The learning rate α, $(0 < \alpha \leq 1)$, is a constant step-size parameter between 0 and 1, and considered as the updating speed of the Q-function. It determines to what extent the newly acquired information will override the old information. A factor of 0 means no learning. A factor of 1 makes responding on the most recent information. The discount factor γ determines the importance of future rewards. A factor of 0 will make an opportunity by only considering current rewards, while γ approaching 1 will make it strive for a long-term high reward. If the discount factor meets or exceeds 1, the Q-value will diverge. The objective is to take the future rewards into the evaluation process more strongly. The Q-function updates its evaluation value of the action while catching into the process of an immediate reinforcement (r) and the Q-value estimation of the new best state $Q^b(\vec{x}(t+1))$. The speed up in Q-learning has been studied in terms of the eligibility trace of an action (Sutton and Barto, 1998). In order to speed up learning, it needs to combine the Q-learning and the temporal method to achieve the eligibility of an action (Jouffe, 1998; Sutton and Barto, 1998).

Fuzzy Q-learning

The basic Q-learning works in the discrete forms of the state-action pairs. As enhanced by fuzzy mathematics, the fuzzy Q-learning can handle continuous state-action pairs (Jouffe, 1998). Fuzzy Q-learning (FQ) algorithm is a model-free reinforcement learning algorithm with an online learning characteristic where the Q-function represents a fuzzy inference system. Define $P = \{p_1, ..., p_N\}$ the set of possible actions of each rule. The Q-learning function is able to choose one among *N* actions

according to their Q-values. Each fuzzy rule is a local representation on input space definitions and memories. The Q-values are related to the selected actions, such that they could maximize the discounted sum of rewards obtained while achieving the task. The fuzzy rules can be presented in the IF-THEN form as in Equation 5, where $x_1, ..., x_n$ are input variables, A_{in} is the fuzzy set, and $\hat{a}_{(i,j)}$ is the output action variable. The learning ability aims to search for the best action for each rule.

$$\begin{aligned} &If\ x_1 \text{ is } A_{i1} \text{ and, . . ., and } x_n \text{ is } A_{in}, \\ &\text{then action } \hat{a}_{(i,1)} \text{ is } p_1 \text{ with } q_{i1} \\ &\qquad\qquad \text{or } p_2 \text{ with } q_{i2} \\ &\qquad\qquad \vdots \\ &\qquad\qquad \text{or } p_n \text{ with } q_{iN} \end{aligned} \quad \text{Rule i}^{\text{th}}: \tag{5}$$

The q_{in} is a Q-value corresponding to an action ($\hat{a}_{(i,j)}$) where *i* is the rule index, and *j* is the possible action index. Assign q_{ii^*} to be the maximum Q-value for rule i by $q_{ii^*} = \max_{1<j<N} q_{ij}$. The greedy action p_{i^*} acquires experience through reinforcement signal (Sutton and Barto, 1998). During the learning process, the action of each rule is based on exploration and exploitation policies (Juang and Lu, 2009). In an exploration process, the non-greedy action is applied to produce a better reward, while the greedy action in an exploitation process. At time step *t*, let $i(t) \in \{1, ..., N\}$ be the identified action chosen by the rule *i*, the system output is given by Equation 6

$$\hat{a}(\vec{x}(t)) = \frac{\sum_{i=1}^{M} \Phi_i(\vec{x}(t)) \cdot p_{i(t)}}{\sum_{i=1}^{M} \Phi_i(\vec{x}(t))} \tag{6}$$

where Φ_i is the firing strength of rule *i*, and M is the total number of rules in the fuzzy system. To identify the practical Q-value for the inference output $\hat{a}$, the expression (7) is used

$$Q(\vec{x}(t), \hat{a}(\vec{x}(t))) = \frac{\sum_{i=1}^{M} \Phi_i(\vec{x}(t)) \cdot q_{ii^*(t)}}{\sum_{i=1}^{M} \Phi_i(\vec{x}(t))} \tag{7}$$

After the action state $\hat{a}(\vec{x}(t))$, at time step *t*, the new state is $\vec{x}(t+1)$. The Q-value is regarded as the best action inferred from the fuzzy inference system. In previous state, for each rule, the greedy action (q_{i^*}) enhances the maximum Q-value q_{ii^*}. Hence, the maximum anticipated Q-value (Q^*) at the state $\vec{x}(t+1)$ is achieved by using Equation 8:

$$Q^*(\vec{x}(t+1)) = \frac{\sum_{i=1}^{M} \Phi_i(\vec{x}(t+1)) \cdot q_{ii^*}}{\sum_{i=1}^{M} \Phi_i(\vec{x}(t+1))} \tag{8}$$

The FQ algorithm can be summarized as follows:

STEP 1: Initialize parameters.
STEP 2: For the current state, compute the membership values of the current state in the neighboring states for all rules.
STEP 3: Compute the current action q_{ii^*} by $q_{ii^*} = \max_{1<j<N}[q_{ij}]$.
STEP 4: Execute the current action and receive a reward γ from the environment.
STEP 5: Compute the current Q-value using (7).
STEP 6: Compute the Q-value of the new state using (8) for each rule.
STEP 7: Choose and update the current state and output action using (6) based on the Q-table model.
STEP 8: Return to STEP 2 for a new trial.

Remarks: (i) the Q-table is constructed offline to be used onboard the controller to save CPU time, and (ii) γ = 0.9 is adopted (Juang and Lu, 2009; Juang and Hsu, 2009; Wakins and Dayan, 1992).

CAR-LIKE ROBOT STRUCTURE

The car-like robot is a testbed agent based on the TLT-1 (Tamiya a little Truck), a one-eighteenth scale radio-controlled (RC) 4×4 Pickup Truck from TAMIYA, Inc. The robot is aimed for an operation in a laboratory with flat terrain. Its loading ability is adequate to support 3 kg-load of various electronic devices including sensors, AVR controller board, ATOM main board and other accessories. The list of the robot's components and associated vendors can be found in Figure 2. Its mechanical specifications are shown in Table 1. The 4 wheel steering gives better moving ability for obstacle avoidance. The robot is recommended for laboratory experiments but robust enough for outdoor operations. Onboard sensors consist of two infrared sensors (GP2Y0D020F), an ultrasonic range finder (SRF05), and a pinhole camera.

The pinhole camera sensor is used for vision purposes, when the car-like robot attempts to avoid the obstacle and find the target. Moreover, the robot applies the camera in various tasks, for instance recognition and obstacle size calculation, obstacle localization, obstacle detection, and route identification. The camera is plugged on the ATOM D510MO main board via a USB interface that provides visual information to the control system. The HIS images from the camera are processed using

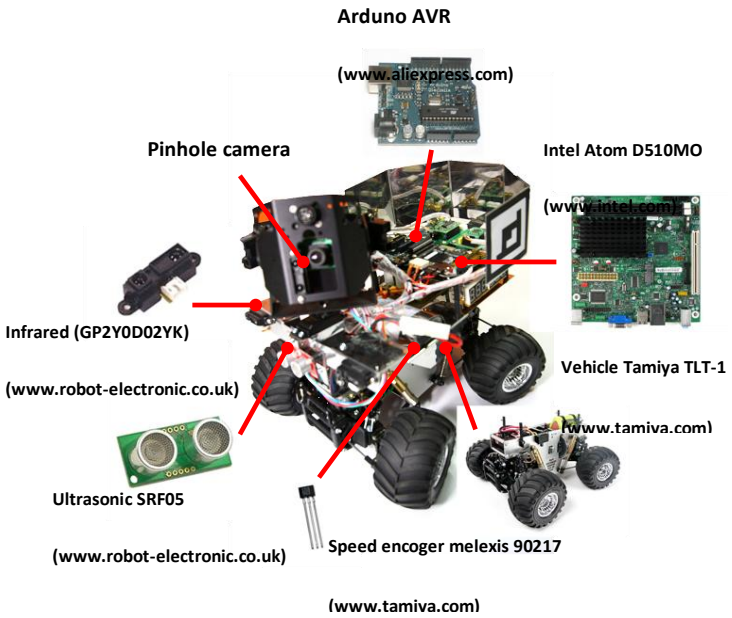

Figure 2. The car-like robot structure.

Table 1. The car-like robot's mechanical specifications.

Dimensions	Length	30 cm
	Width	22 cm
	Height	25 cm
Weight	Load (Max)	3 kg
	Unload	1.5 kg
Max speed		1.6 m/s
Max torque		55 N/m
Carry capacity		2.5 kg
Min steering radius	4 wheel steering	30 x 30 cm^2

MATLAB for RGB colour filtering and edge detections. The sampling time of the vision system is chosen sufficiently higher than the average image processing time interval. The information received from the pinhole camera is always available on time. The sampling time of the vision system is set at 250 ms, while the sampling time of the control system is set at 25 ms. Two infrared (IR) arrays (GP2Y0D020F, 24 cm range) are mounted on the front part of the robot to detect the obstacle range. Each IR range sensor produces an output in binary format proportional to the detected distance against the obstacle. The IR range sensor transmits the distance information via an I^2C bus in real time. One ultrasonic range finder (SRF05) is mounted on the front part of the robot. This sensor offers a measurement range of 0 to 3 m with a wide viewing angle of 60° beaming. The measurement range is set at 0 to 1.5 m for the car-like robot.

CONTROL DESIGN

Figure 3 represents the robot control system. Motion control of the robot at low level is accomplished by the ordinary PI control of 3 DC motors, one of which is dedicated to forward and backward motion. The other 2 motors are for turning of the front and rear wheels, respectively. To achieve an environmental learning capability, the system employs visual feedback via a pinhole camera with conventional colour and edge detection algorithms such that the target and the obstacles could be recognized. The image processing

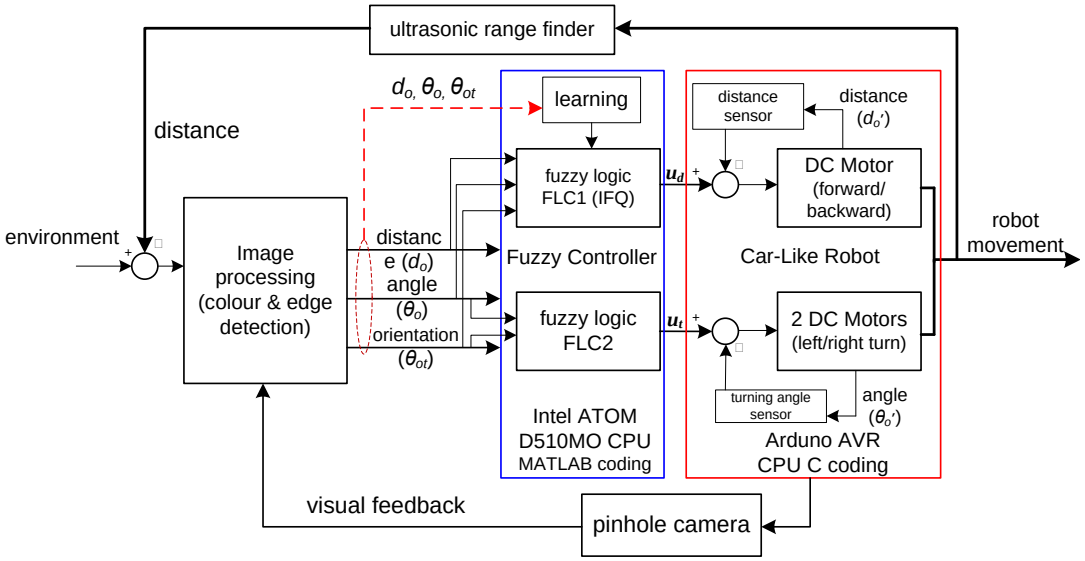

Figure 3. Car-like robot control system.

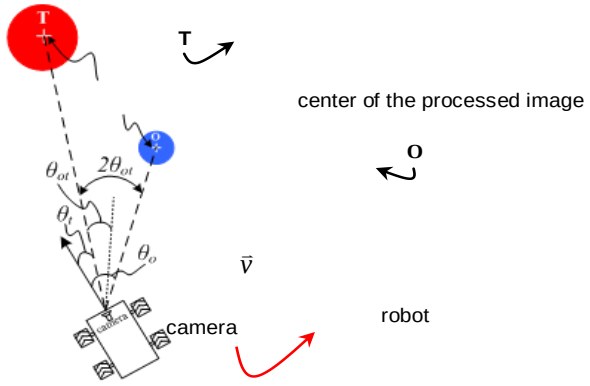

Figure 4. Navigation of the robot.

results in distance and angles information to be used by the IFQ-learning control. An ultrasonic range finder is also used to ensure the correctness of the distance measurement.

Object detection: There are two types of objects in the scenario. A red cylinder with 40 cm diameter represents the target. Several blue cylinders each having 20 cm diameter are obstacles. An image captured is processed using conventional colour coding on RGB and HSI formats as well as edge detection. The image is updated every 250 ms, and processed on an Intel ATOM D510MO board. For the robot to begin its motion, it firstly performs a counter-clockwise turnaround to seek for the target. The distance between the target and the robot is measured by an ultrasonic range finder. A captured image also contains the pictures of obstacles. Using a similar approach, the robot can recognize the obstacles. The ultrasonic sensor also provides the distance information between the robot and the obstacles.

Obstacle avoidance: Once the target is found, the robot attempts to move towards it in a straight path at a constant speed of 18 m/min. It stops by the target with a 40 cm clearance distance. Along the travelling course the robot may encounter some obstacles. Under the control by a learning FLC, it may either turn away at once or move backward and turn away from the obstacle. To navigate the robot towards the target needs to know the orientation angle. Referring to Figure 4, the target (T) is always placed at the center of the processed image. $\vec{v}$ represents the velocity and the current direction of the robot. θ_0 and θ_t represent the measured angles between and the target and the nearby obstacle (O), respectively.

With such information, the orientation angle (θ_{ot}) can be obtained, that is, $\theta_{ot} = (\theta_0 - \theta_t)/2$. The plus and minus signs represent right and left orientation, respectively.

Fuzzy control rules

The proposed learning control employs fuzzy rules as the robot navigation aid. There are two groups of rules namely towards the target or FLC1, and robot turning or FLC2. The fuzzified input variables are distance measure, obstacle orientation to the robot and orientation angle (θ_{ot}), respectively. The fuzzified output variables are robot moving distance and turning command, respectively. Figure 5 shows the membership functions of these variables. The design of the fuzzy rules is heuristic, and employs the maximum height inference technique.

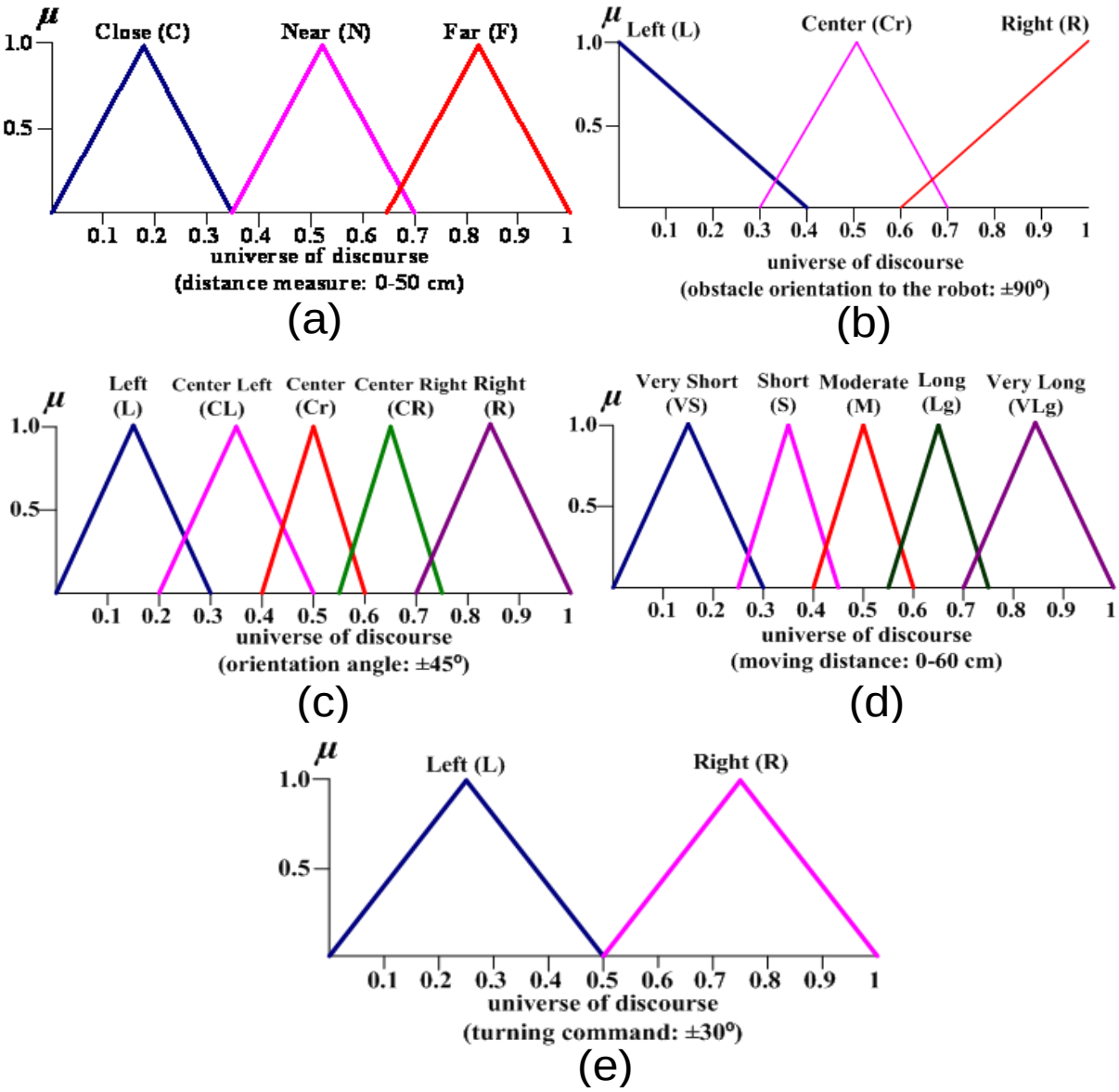

Figure 5. Membership functions of input and output variables: input variables (a) distance measure, (b) obstacle orientation and (c) orientation angle (θ_{ot}); output variables (d) moving distance and (e) turning command.

The FLC1 composes of 3 sub-groups according to the following rules:

i) if (obstacle is close) then (temporarily stop, execute sub-group 1 rules).
ii) if (obstacle is near) then (keep moving in the same direction and execute sub-group 2 rules)
iii) if (obstacle is far) then (temporarily stop, execute sub-group 3 rules).

Rules of sub-group 1: Summary of the rules is presented in Table 2. A rule can be read as "if (obstacle orientation is left) and (angle of orientation is right) then (move backward with a moderate distance)", for instance.

Table 2. Sub-group 1 rules of FLC1.

θ_o \ θ_{ot}	L	CL	Cr	CR	R
L	M	Lg	VLg	Lg	M
Cr	M	Lg	VLg	Lg	M
R	M	Lg	VLg	Lg	M

Rules of sub-group 2: Table 3 summarizes the rules. As an example, a rule can be read as "if (obstacle orientation is center) and (angle of orientation is left) then (move forward with a very short distance) and (call FLC2)". Rule of sub-group 3: The rule summary is in Table 4. A rule

Table 3. Sub-group 2 rules of FLC1.

θ_o \ θ_{ot}	L	CL	Cr	CR	R
L	M	S	VS	S	M
Cr	VS	S	M	S	M
R	M	VS	S	VS	M

Table 4. Sub-group 3 rules of FLC1.

θ_o \ θ_{ot}	L	CL	Cr	CR	R
L	VLg	Lg	M	Lg	VLg
Cr	Lg	Lg	M	Lg	VLg
R	VLg	Lg	M	Lg	VLg

can be read as "if (obstacle orientation is right) and (angle of orientation is left) then (move forward with a very long distance)", for instance.

The FLC2 for turning of the robot consists of a few rules as follows:

i) If (obstacle orientation is left) then (turn right),
ii) If (obstacle orientation is center) then (turn right), and
iii) If (obstacle orientation is right) then (turn left).

FQ-learning control

Robot control algorithms are represented by the flow diagram in Figure 6 in which the switch (sw) is at (a) for the FQ-learning, while at (b) for the IFQ-learning.

The microcontroller onboard the robot initializes the states, and successfully recognizes the positions of the target and the obstacles using the conventional image processing techniques. The robot starts moving according to the FLC1, and executes the FQ-algorithm according to the 8 steps described in "Learning control". Assume that the initial $Q^a\{\bar{x}(t), a(t)\} = Q\{3,3\}$ = 0.5, α = 0.5, γ = 0.9, $r(t+1)$ = -0.04 and $Q^b(\bar{x}(t+1))$ = 0.11, the Q value can be calculated using Equation 3 as follows:

1st: $Q(\rightarrow,(3,3))$ = 0.5+0.5(-0.04+0.99-0.5) = 0.725

2nd: $Q(\rightarrow,(3,3))$ = 0.725+0.5(-0.04+0.99-0.725) = 0.838

3rd: $Q(\rightarrow,(3,3))$ = 0.838+0.5(-0.04+0.99-0.838) = 0.894

⋮

nth: $Q(\rightarrow,(3,3))$ = 0.947+0.5(-0.04+0.99-0.947) = 0.947.

The value of Q = 0.947 is mapped to a rule of each sub-group in order to active a better performance of the FLC1. Such mapping is drawn from the robot training phase. To reduce the CPU burden during real-time operation, the Q values are calculated offline for the rules of each sub-group of the FLC1, and stored in the onboard memory as a table denoted as the Q-table. The robot continues moving unless an obstacle blocks its path. The FLC2 is called for the robot to turn away from the obstacle if its path is blocked. The functions are repeated until the robot reaches the target.

IFQ-learning control

Referring to Figure 6, the switch (sw) is at (b). The immune algorithm described in "Learning control" is incorporated into the robot control. After the robot encounters an obstacle, moves backward and retries, the immune algorithm adjusts the vertices of the membership functions of the distance measure variable (Refer to STEPs 2 to 9 in "Fuzzy Q-learning"). This mechanism is represented by the diagram in Figure 7. The tuned membership functions are regarded as the clones of the original ones. As an example, before cloning the vertices of the close, near and far are at 0.175, 0.525 and 0.825, respectively. After cloning, they may be shifted to 0.275, 0.475 and 0.775, respectively.

The immune algorithm stated in "Artificial Immune System (AIS)" is combined with the FQ-learning algorithm stated in "Fuzzy Q-learning". The cooperative algorithm is denoted as the immune fuzzy-Q learning or IFQ algorithm. The immune algorithm helps adjust the membership functions of fuzzy descriptors in an online manner subject to environment changes. It is expected that the proposed IFQ algorithm can learn and act quickly to the problems and changes. The proposed algorithm is listed as follows:

Proposed IFQ algorithm

STEP 0: Initialize parameters: *Ag, Ab*$_{j,k}$, *Ag*$_{i,k}$, *Abn*, *C*, *D*, *IR_left*, *IR_right, M*, *Mc, Mc*$_{cand}$, *AT, C**, *Cd*, *C'*, *Nd*, *N'*, *Fd*, *F'*, δ , θ_o , θ_{ot}, $q_{ii}{}^*$, and Φ_i .

STEP 1: Normalize information antigens (*Ags*) of d_0, θ_o , and θ_{ot} in the Euclidian distance range [0, 1] and calculate distances between *Ags* and *Abs* using (1).

STEP 2: Create memory pool (*M*) in matrix form.

STEP 3: For each antigen *Ag*$_i$ from the antigen population *Ag*, calculate affinity vector according to input for all antibodies using

$$\text{affinity}(Ag, Ab) = 1/\left[1+\sqrt{\sum_{i=1}^{L}\left|Ag_i - Ab_i\right|}\right]$$

$$Ab_i = \frac{nc_i}{S}$$; *i* = 1,2,…,*L*, and *L* is a real number.

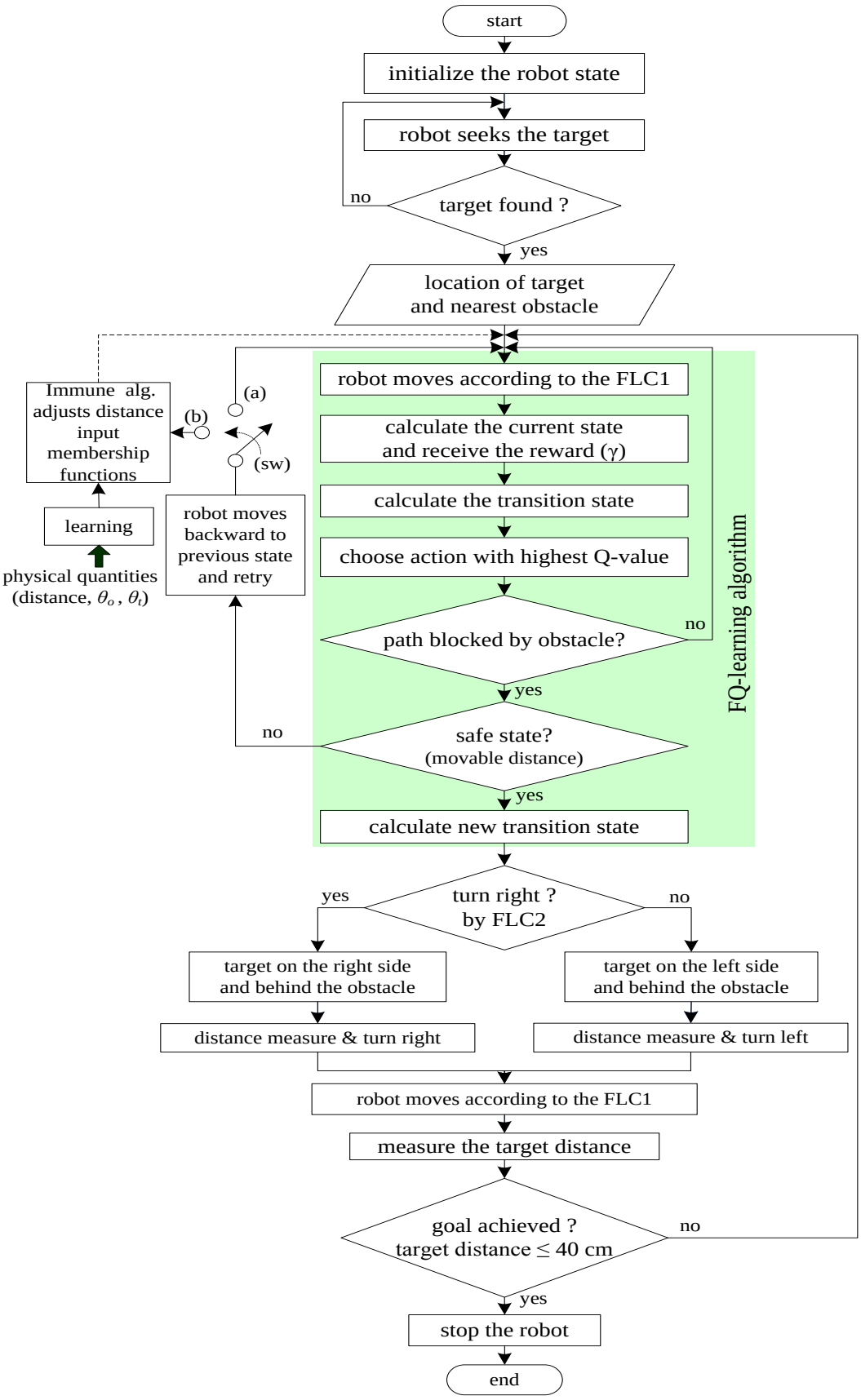

Figure 6. Flow diagram of robot control.

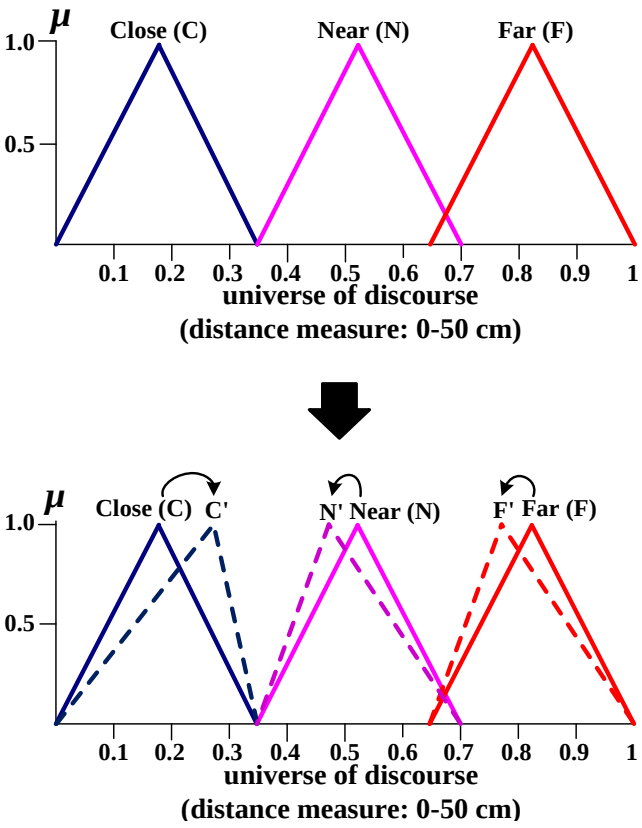

Figure 7. Membership functions tuning by immune algorithm.

STEP 4: Select highest affinity antibodies from *Abs* to compose a set *Abn*.
STEP 5: Randomly generate a set *C* of clones.
STEP 6: Generate a mutated antigen *Ag* in set *C**
STEP 7: Calculate affinity threshold AT using (2).
STEP 8: Update memory pool (M)

If $|Mc_{cand}| < AT$ then
If M is not filled up then store Mc_{cand} in M.
If M is filled up then replace the oldest Mc_{cand} by the current Mc_{cand}.
else
go to STEP 9.
endif

STEP 9: Check the signal from the infrared sensor

If (*IR_left* or *IR_right*) = "1"
then update vertices of input distance member functions: $C' = Cd + C$, $N' = Nd + C$ and $F' = Fd + C$;
go to Step 5.
else (*IR_left* and *IR_right*) = "0"
then go to STEP 10.
endif

STEP 10: For the current state, compute the membership values of the current state in the neighboring states for all rules.
STEP 11: Compute $q_{ii^*} = \max_{1<j<N}[q_{ij}]$.
STEP 12: Execute the current action and receive a reward γ from the environment.
STEP 13: Compute the current Q-value using (7).
STEP 14: Compute the Q-value of the new state using (8) for each rule.
STEP 15: Choose and update the current state and the output action using (6) based on the Q-table model.
STEP 16: Go to STEP 10 for a new trial.

Remarks: (i) the Q-table is constructed offline to be used onboard the controller to save CPU time, and (ii) $\gamma = 0.9$ is adopted (Juang and Lu, 2009; Juang and Hsu, 2009; Wakins and Dayan, 1992).

EXPERIMENTS

The effectiveness of the proposed IFQ algorithms for controlling a car-like robot is illustrated. The task of the robot is to reach the target while avoiding obstacles. The robot must be able to cope with environmental changes, that is, configurations of the obstacles and the target change. The robot seeks the target, designs its path to the target then moves directly towards the target as the shortest path. When the robot senses obstacles directly in front, it makes a turning action. If the center of obstacle gravity, CGO, is on the left side of the center of target gravity, CGT, and the target is behind the obstacle, the robot turns right (Figures 8a and 8d). On the contrary, the robot turns left if the CGO is on the right side (Figures 8b and 8e). The robot moves straight according to no presence of an obstacle (Figures 8c and 8f).

During the training phase, the robot moves and turns under supervision. The goal is to achieve an adjustment of the membership functions, and a smoothed trajectory. The training scenarios are designed for left turns, right turns, and straight moves with the different sizes of obstacles. If a blocking obstacle is found, the transmitted image from the camera system is used for extracting the edges, the CGO and the CGT. In this case, the fuzzy control rules make decision for appropriate turning direction and distance. Once the robot approaches the target, it will stop with a clearance of 40 cm (Figure 8f).

The illustrations in Figure 9 represent 9 experiments under 5 different obstacle setups. The blue and red cylinders represent the obstacles and the target, respectively.

The robot experiences different situations that cause turning problems regarding sizes and locations of the obstacles. In the first experiment, an obstacle is directly in front of the robot, and the target is located behind the obstacle for various start positions A, B, and C (Figure 9a). In the second situation, the robot faces two attached obstacles for the start position A but no obstacle for B and C (Figure 9b). In the third situation, three obstacles are arranged to block any straight path to the target (Figure 9c-f). In the fourth situation, the robot is challenged with two turning behaviors imposed by four obstacles as shown in Figure 9g for the start positions A, B, and C. In the fifth situation, the robot faces more challenging tasks as shown in Figure 9h-i. These five situations are designed to study the comparative effectiveness of the fuzzy control, the FQ- and the IFQ-learning control algorithms under complicatedly unknown environments.

Note that, these comparison studies use the same fuzzy rules. According to the IFQ algorithm, the robot keeps adjusting its turning distance every time it navigates a new environment. Hence, the learned information keeps evolving all the time. Chronologically experienced information is removed every 30 min by the "first-in-first-out" concept. To reduce computational burden the Q-value table is used instead of online Q-learning calculation.

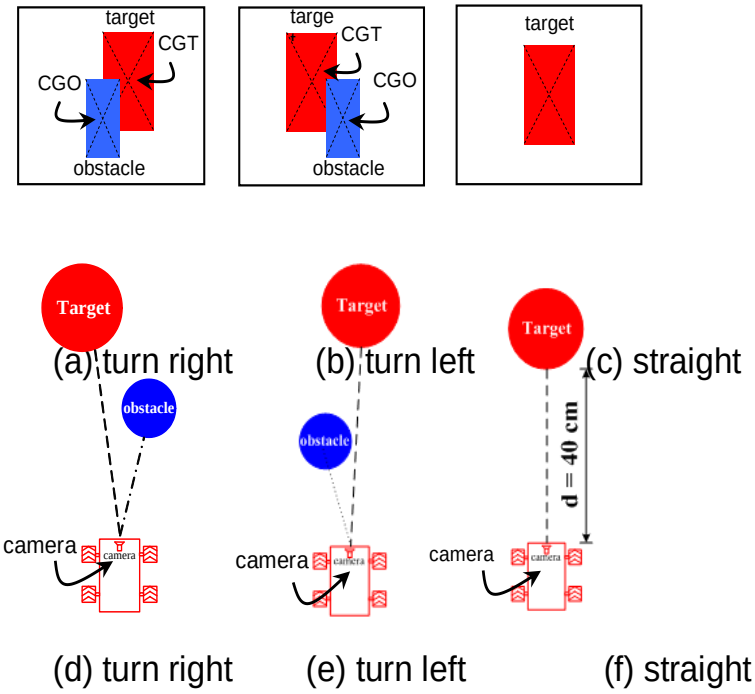

Figure 8. Basic robot turning rules.

RESULTS AND DISCUSSION

The robot motion is kept at a constant speed of 18 m/min. Four-wheel turning is used to achieve 30° left/right turns. Regarding previous experiences, the more training the robot gets, the better performances the robot has and hence the success rate improves. In this research, the robot is trained using 30 different setups with random starting positions to study the effectiveness of the control algorithms. The effects of the obstacle sizes, the number of obstacles and the starting positions of the robot are studied to compare the control algorithms on the rates of retry turning and goal achievement.

Figure 9a depicts the simplest obstacle arrangement with 3 different robot starting positions (A to C). In this experiment 1, the robot employs the simple fuzzy control rules and the FQ-learning rules. For all starting positions, the robot fails with a few turning retries before achieving a successful path to reach the target. From observation, the FQ-learning rules assist the robot on memorizing the failed paths and not repeating them. This is not the case with the fuzzy control that produces success rates of 40 to 60%.

Figure 9b depicts the second arrangement of the target-obstacle map in which 2 obstacles are attached to form one large obstacle regarded as obstacle #1. The obstacle #1 blocks a direct path of the robot from the start position A to the target. With the fuzzy control and the FQ-learning, the robot starting at the B and C positions immediately finds direct paths to the target as shown in Figure 10a. However, it fails to avoid colliding with the obstacle #1 after leaving the start position A (Figure 10a). The robot attempts several retries as shown in Figure 10b and c, but still fails. The reasons are that the fuzzy control cannot adapt to the environment changes, and the FQ-learning inadequately covers the environment changes, particularly the obstacle size in this case. The same target-obstacle map is applied further for the robot embedded with the IFQ-learning rules. When the robot senses the obstacle #1 blocking the desired path, the immune algorithms adjust the fuzzy turning rules by adding cloning parameters. In effect, the robot realizes more spaces for turning, and eventually seeks its way to avoid colliding with the obstacle to reach the target. Figure 11 represents the robot paths in this case (VDO-clip available at http://www.sut.ac.th/engineering/electrical/carg/robot.htm)

Figure 9c-i depicts different target-obstacle maps for more difficult experiments designated as experiments 3 to 9 accordingly. Relative distances among the obstacles and the target raise the levels of difficulty of the problems. From these pictures, the readers can notice different paths made by the robot to reach the target.

Figure 9c-d represent similar situations in a symmetrical form. A large obstacle #1 formed from two attached obstacles and a small obstacle #2 block a direct path from the start position A to the target. Another small obstacle #3 imposes a constraint on a possible route to the target. With the fuzzy control and the FQ-learning, the robot completely fails to avoid the obstacle #1 as indicated

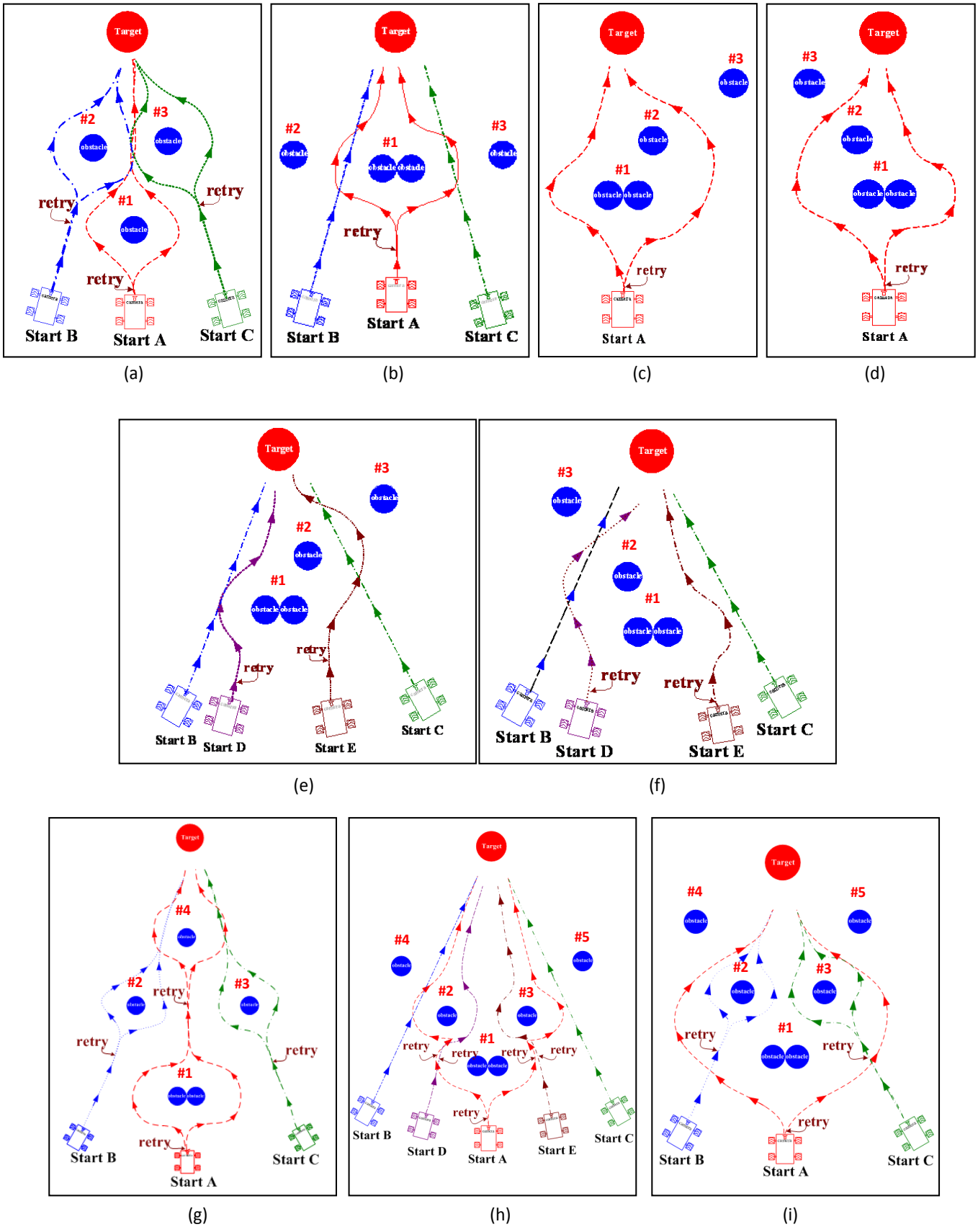

Figure 9. Experimental target-obstacle maps: (a) Experiment 1 (b) Experiment 2, (c) Experiment 3, (d) Experiment 4, (e) Experiment 5, (f) Experiment 6, (g) Experiment 7, (h) Experiment 8 and (i) Experiment 9.

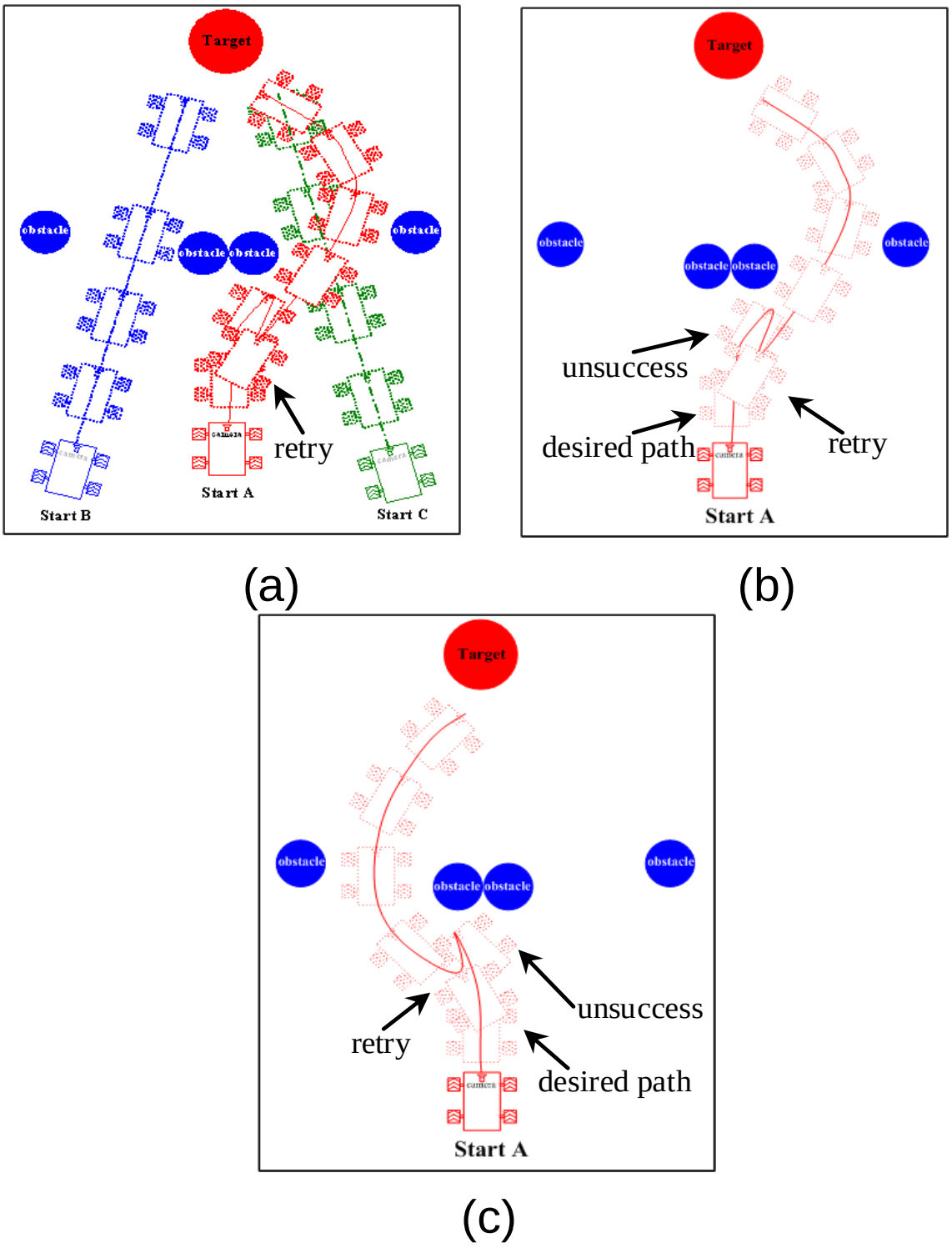

Figure 10. Path tracking by using fuzzy control and FQ-learning algorithm (a) start positions A, B and C, (b) start position A and right turn, and (c) start position A and left turn.

in Table 5 of the Experiments 3(3) and 4(3). The robot retries once to successfully recognize a large obstacle and turn away from it under the proposed IFQ-learning. The situations are shown in Figure 9c-d with the summary in Table 5. The chance for the IFQ-learning control to decide for left or right turning is

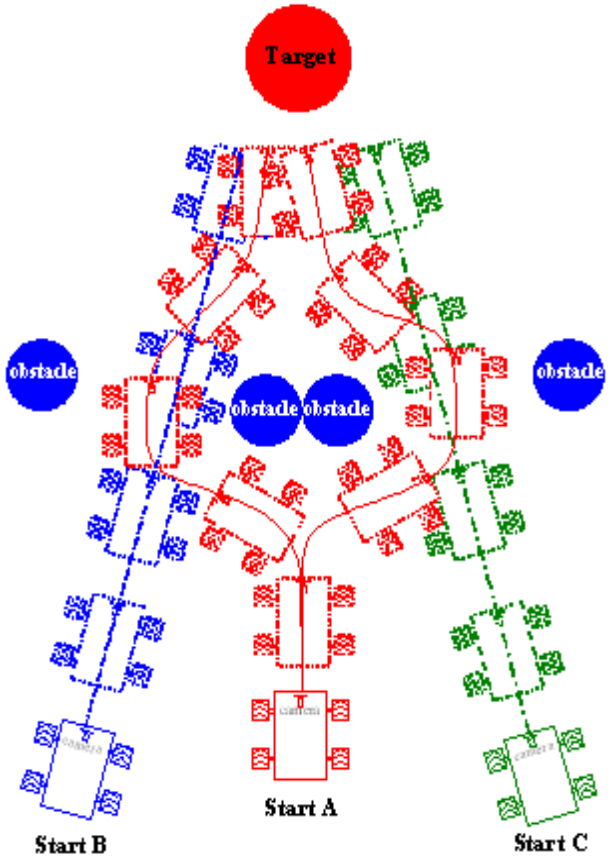

Figure 11. Diagram representing robot behaviors with the IFQ-learning.

approximately equal. Similar experiments designated as 5 and 6 are also conducted with 4 different start positions. The situations are represented by the pictures shown in Figure 9e-f. With the fuzzy control and the FQ-learning, the robot only at the start positions B and C successfully reaches the target. Again, it partially fails to recognize a large obstacle #1 when this obstacle is directly in front of the traveling path (the robot at the start positions D and E). With the IFQ-learning, the robot successfully avoids all obstacles and reaches the target without any retry. The data in Table 5 for the experiments 5(3) and 6(3) summarize such results.

Figure 9g depicts one complicated target-obstacle map designated as experiment 7, and 7(4) in Table 5 accordingly. With the proposed IFQ-learning, the robot at the start positions A, B, and C successfully travels to the target and avoids colliding with all obstacles without any retry. In contrast, the robot with the FQ-learning needs some retries to recognize and avoid the large obstacle #1. When the start position is at A, the success rate is only 20% with the FQ-learning, but the robot completely fails with the simple fuzzy control. The success rates become 40% for the robot with the fuzzy control when the start positions are at B and C.

More complicated experiments are represented by the pictures in Figure 9h-i. There are 4 small obstacles and a large one. The relative distances among these obstacles and the target are set differently. Referring to Figure 9h and experiment 8(5) in Table 5, the robot with the fuzzy control and the FQ-learning at the start positions B and C completely reaches the target without any retry. This is not the case for the start positions A, D and E, in which the robot completely or partially fails to reach the target. With the proposed IFQ-learning, from all the start positions the robot successfully moves to the target without any retry except that 2 retries are performed at the start position A. Referring to Figure 9i and experiment 9(5) in Table 5, similar results are observed. Without the immune algorithm, it is difficult for the robot to learn a changed environment.

The experimental results discussed so far are summarized in Table 5. With the fuzzy control and the FQ-learning, the success rates of the robot reaching the target are 20 to 40% or null when the robot faces a large obstacle blocking a direct path to the target. The reasons are that the simple fuzzy control does not have adaptive mechanism, hence cannot learn or adapt to environment changes, and the FQ-learning algorithm, although works well to some extent, is not adequate to cope with complicated environment changes. In the contrary, the proposed IFQ-learning provides 100% success rates with only a few retries. Moreover, the robot consumes a shorter traveling time to the target indicated by the average time values in comparison with the other approaches. The artificial immune algorithm assists the fuzzy controller to adapt to environment changes by adjusting the membership functions. As a result, the IFQ controller decides for the robot to turn away from the obstacles with longer turning distances, and hence not hitting the obstacles. The experimental results confirm the effectiveness of the proposed IFQ-learning algorithms.

Conclusion

This article has proposed an artificial immune fuzzy Q-learning controller for obstacle avoidance and target tracking of an autonomous car-like robot. Simple fuzzy control rules are employed to supervise the low-level PI control loops of 3 DC servo motors for driving and turning the robot. The fuzzy rules are firstly enhanced by the Q-learning technique denoted as the FQ-learning control. Secondly, the artificial immune algorithms have been incorporated into the controller to achieve a new strategy denoted as the IFQ-learning control. The article has given the details of these algorithms and control design.

Experimental setup has been constructed on a TLT-1 (Tamiya a little truck). The car-like robot is equipped with various sensors including the pinhole camera to provide visual feedback. It uses an AVR controller board with the ATOM D510MO. Nine complicated experiments have been conducted to investigate the effectiveness of the

Table 5. Summary of experimental results.

Experiment (number of obstacles)	Start position	Fuzzy control					Fuzzy Q-learning					Immune Fuzzy Q-learning				
		Trial (times)	Retry (times)	Success (times)	Success rate (%)	Average time (s)	Trial (times)	Retry (times)	Success (times)	Success rate (%)	Average time (s)	Trial (times)	Retry (times)	Success (times)	Success rate (%)	Average time (s)
1 (3)	A	5	5	2	40	28.33	5	2	4	80	19.47	5	0	5	100	17.56
	B	5	5	3	60	25.15	5	2	4	80	18.11	5	0	5	100	18.00
	C	5	5	2	40	27.42	5	2	4	80	18.35	5	0	5	100	17.55
2 (3)	A	5	5	0	0	∞	5	5	0	0	∞	5	1	5	100	34.24
	B	5	0	5	100	17.54	5	0	5	100	17.22	5	0	5	100	17.33
	C	5	0	5	100	19.44	5	0	5	100	18.39	5	0	5	100	18.11
3 (3)	A	5	5	0	0	∞	5	5	0	0	∞	5	1	5	100	36.15
4 (3)	A	5	5	0	0	∞	5	5	0	0	∞	5	1	5	100	40.53
5 (3)	B	5	0	5	100	18.40	5	0	5	100	17.55	5	0	5	100	17.45
	C	5	0	5	100	17.29	5	0	5	100	17.41	5	0	5	100	16.75
	D	5	5	0	0	∞	5	2	4	80	31.05	5	0	5	100	18.38
	E	5	5	0	0	∞	5	5	0	0	∞	5	0	5	100	21.15
6 (3)	B	5	0	5	100	17.30	5	0	5	100	16.44	5	0	5	100	16.53
	C	5	0	5	100	17.47	5	0	5	100	17.01	5	0	5	100	16.72
	D	5	5	0	0	∞	5	5	0	0	∞	5	0	5	100	25.04
	E	5	5	0	0	∞	5	1	4	80	28.30	5	0	5	100	19.49
7 (4)	A	5	5	0	0	∞	5	5	1	20	73.20	5	0	5	100	32.10
	B	5	3	2	40	21.23	5	1	5	100	16.53	5	0	5	100	17.02
	C	5	3	2	40	19.21	5	1	5	100	17.26	5	0	5	100	17.32
8 (5)	A	5	5	0	0	∞	5	5	0	0	∞	5	2	5	100	38.16
	B	5	0	5	100	17.23	5	0	5	100	17.05	5	0	5	100	17.21
	C	5	0	5	100	17.00	5	0	5	100	18.29	5	0	5	100	17.72
	D	5	3	1	20	22.40	5	1	2	40	24.34	5	0	5	100	20.30
	E	5	2	2	40	24.11	5	1	2	40	28.10	5	0	5	100	19.55
9 (5)	A	5	5	0	0	∞	5	5	0	0	∞	5	1	5	100	37.22
	B	5	4	2	40	17.22	5	1	5	100	18.34	5	0	5	100	16.47

fuzzy, the FQ- and the IFQ-learning controllers for comparison purposes. The setups are illustrated in "Experiments" with some VDO-clips available at http://www.sut.ac.th/engineering/electrical/carg/robot.htm. As a result, the success rates of the robot reaching the target are 43.70% as average using the fuzzy control, and 62.96% as average using the FQ-learning control, while 100% success rates are achieved via the proposed IFQ-learning approach. Applying such algorithms to a convoy of robots will be our future research issues.

ACKNOWLEDGEMENTS

This work was supported by Suranaree University of Technology (SUT) and by the Office of the Higher Education Commission under NRU project of Thailand.

Notation list

a(t) = The action at time state (*t*)
Ab = An antibody
Abn = The set of antibodies with high affinities
Abs = Antibodies
Ab_i = The attribute of antigen *(i = 1,…, Mc)*
$Ab_{i,k}$ = The k^{th} attributes of the Ab_i
A_{in} = Fuzzy set
Ag = An antigen
Ags = Antigens
Ag_i = The attribute of antigen *(i = 1,…,N)*
$Ag_{i,k}$ = The k^{th} attributes of the Ag_i
AT = The affinity threshold
C = The set of clones
*C** = The mutated antigen
C' = The vertex parameter of triangular membership function describing close after adding the set of clones
Cd = The vertex parameter of triangular membership function describing close
D = The distance between *Ags* and *Abs*
F' = The vertex parameter of triangular membership function describing far after adding the set of clones
Fd = The vertex parameter of triangular membership function describing "far"
IR_left = The infrared distance finder signal (left)
IR_right = The infrared distance finder signal (right)
M = The memory pool
Mc = The number of memory
Mc_{cand} = The candidate memory
nc_i = The number of recognized patterns to be classified in the i^{th} subset
N' = The vertex parameter of triangular membership function describing near after adding the set of clones
Nd = The vertex parameter of triangular membership function describing "near"
$r(t+1)$ = the scalar reinforcement signal that depends on environment
x_n = The fuzzy input variables
$\vec{x}(t)$ = The present state
$\vec{x}(t+1)$ = The obtained state from $\vec{x}(t)$ after the current action of *a(t)*
$\hat{a}$ = The inference output
$\hat{a}(\vec{x}(t))$ = The action state at time step *t*
Q^* = The maximum anticipated Q-value
$Q(\vec{x}, a)$ = The output for state vector $\vec{x}$ and action (*a*)
$Q(\vec{x}(t+1))$ = The possible action sets at the state $\vec{x}(t+1)$
$Q^b(\vec{x}(t+1))$ = The best estimated Q-value that the agent assumes to reach the state $\vec{x}(t+1)$
S = The antibodies space
$\hat{a}_{(i,\ j)}$ = The output action variable
δ = The average distance between each Ag_i in *Ags*
α = The learning rate, $(0 < \alpha \leq 1)$
γ = The discount factor
θ_t = The target angle
θ_o = The obstacle angle
θ_{ot} = The orientation angle
n = The highest affinity antibodies from *Ab*
p_{i*} = The greedy action
P = The set of possible actions of each fuzzy rule
q_{in} = The Q-value corresponding to an action
q_{ii*} = The maximum Q-value for rule *i*
Φ_i = The firing strength of rule *i*

REFERENCES

Abbas AK, Lichtman AH (2000). Cellular and molecular immunology. Elsevier Saunders, US.
Albus JS (1991). Outline for a theory of intelligence. IEEE Trans. Syst. Man. Cybern. C Appl. Rev. 21(3):432-509.
Amin SHM, Adriansyah A (2006). Particle swarm fuzzy controller for behavior-based mobile robot. In Proceedings of an IEEE International Conference on Control, Automation, Robotics and Vision (ICARCV'06), Singapore. pp. 1-6.
Antaklis PJ (1994). Defining intelligent control. IEEE Contr. Syst. Mag. 14(3):58-66.
Castro de LN, Timmis J (2002). Artificial immune systems:A new computational intelligence approach. Springer, UK.
de Castro LN, Von Zuben FJ (2002). Learning and Optimization Using the Clonal Selection Principle. IEEE Trans. Evol. Comput., Spec. Issue Artif. Immune Syst. (IEEE) 6(3):239-251.
Er MJ, Deng C (2004). Online tuning of fuzzy inference systems using dynamic fuzzy Q-learning. IEEE Trans. Sys. Man. Cybern. B Cybern. 34(3):1478-1489.
Farrell J, Baker W (1995). Learning control systems:motivation and implementation. IEEE Press, US.
Garbi GP, Orlando V, Rosado G, Gradinetti FJ (2007). Multivalued adaptive neuro-fuzzy controller for robot vehicle. In Proceedings of an International Conference on Intelligent Systems, and Knowledge Engineering (ISKE07). Chengdu, China. pp. 883-891.
Hagras H, Callaghan V, Colley M (2000). Online learning of the sensors' fuzzy membership functions in autonomous mobile robots. In Proceedings of an IEEE International Conference on Robot and

Automation, San Francisco. pp. 3233-3238.
Jerne NK (1974). Towards a network theory of the immune system, Ann. Immunol. 125C:373-389.
Jouffe L (1998). Fuzzy inference system learning by reinforcement methods, IEEE Trans. on Syst. Man. Cybern. C Appl. Rev. 28(3):338-355.
Juang CF, Hsu CH (2009). Reinforcement ant optimized fuzzy controller for mobile-robot wall-following control. IEEE Trans. Ind. Electron. 56(1):3931-3940.
Juang CF, Lu CM (2009). Ant colony optimization incorporated with fuzzy Q-learning for reinforcement fuzzy control. IEEE Trans. Sys. Man. Cybern. A Syst. Hum. 39(3):597-608.
Knight T, Timmis J (2003). A multi-layered immune inspired machine learning algorithm. Appl. Sci. Soft. Comput. 2:195-220.
Li G, Zhuang J, Hou H, Yu D (2009). An improved clonal selection classifier incorporating fuzzy clustering. IEEE Int. Conf. Meas. Technol. Mechatron. Automat. 3:179-182.
Parhi DR, Sing MK (2009). Navigational path analysis of mobile robots using an adaptive neuro-fuzzy inference system controller in a wdynamic environment. Proc. Inst. Mech. Eng. C: J. Mech. Eng. Sci. 224(6):1369-1381.
Rusu P, Petriu EM, Whalen TE, Cornell A, Spoelder HJW (2003). Behavior-based neuro-fuzzy controller for mobile robot navigation. IEEE Trans. Instrum. Meas. 29(5):1335-1340.
Senthilkumar KS, Bharadwaj KK (2009). Hybrid genetic-fuzzy approach to autonomous mobile robot. In Proceedings of an IEEE International Conference on Technologies for Practical Robot Application (TePRA), Woburn, US. pp. 29-34.
Sutton RS, Barto AG (1998). Reinforcement Learning. MIT Press, US. pp. 4-6.
Wakins CJ, Dayan P (1992). Q-learning. J. Mach. Learn. 8(3):279-292.
Xu WL, Tso SK (1999). Sensor-based fuzzy reactive navigation of mobile robot through local target switching. IEEE Trans. Syst. Man. Cybern. C Appl. Rev. 29(3):451-459.
Zhu A, Yang SX (2007). Neuro-fuzzy-based approach to mobile robot navigation in unknown environments. IEEE Trans. Sys. Man. Cybern. C Appl. Rev. 37(4):610-621.
Zhuang GL, Hou JH, Yu D (2009). An improved clonal selection classifier incorporating fuzzy clustering, In Proceedings of an IEEE International Conference on Measurement Technology. Mechatronic Autom. pp. 179-182.