# Q-learning using Fuzzified States and Weighted Actions and Its Application to Omni-direnctional Mobile Robot Control

Dong-Hyun Lee, In-Won Park and Jong-Hwan Kim

*Abstract*— **The conventional Q-learning algorithm is described by a finite number of discretized states and discretized actions. When the system is represented in continuous domain, this may cause an abrupt transition of action as the state rapidly changes. To avoid this abrupt transition of action, the learning system requires fine-tuned states. However, the learning time significantly increases and the system becomes computationally expensive as the number of states increases. To solve this problem, this paper proposes a novel Q-learning algorithm, which uses fuzzified states and weighted actions to update its state-action value. By applying the concept of fuzzy set to the states of Q-learning and using the weighted actions, the agent efficiently responds to the rapid changes of the states. The proposed algorithm is applied to omni-directional mobile robot and the results demonstrate the effectiveness of the proposed approach.**

## I. INTRODUCTION

Q-learning is one of the popular reinforcement learning methods, which uses state-action value. This state-action value represents the expected return of state-action pair and agent can generate an optimal policy by updating this value. Although Q-learning method does not require any supervisor and creates the optimal policy from experience, agent can only perform one action in one state. This may cause an abrupt transition of action as the state radically changes. As an example, if the conventional Q-learning is used to control a system, which requires smooth and continuous manipulation, the system requires a large number of discretized states and discretized actions. This will cause long training time for the controller to generate the optimal policy.

There were various researches to solve this problem, such as fuzzy Q-learning and model-based reinforcement learning. Adaptation schemes of Q-learning were presented for fuzzy inference systems [1]-[4]. Fuzzy approximation structure for Q-learning was developed and combined with the model-based $Q$ value iterative algorithm [5], [6]. The fuzzy reinforcement learning was applied to power management and control in wireless transmitters [7], [8]. The fuzzy Q-learning was applied to robot navigation [9] and the fuzzy reward function was presented for improving the learning efficiency [10].

To complement the weakness, fuzzified states and weighted actions are applied to the conventional Q-learning method in this paper. Fuzzified states are defined by the fuzzy sets in fuzzy logic control. Fuzzy logic control uses fuzzy sets to represent the environment and controls the system with continuous outputs. However, it requires fuzzy rules, which should be designed by experts.

This paper proposes Q-learning method which uses fuzzified states and weighted actions. By using the fuzzy representation to the states of Q-learning method, the system can generate a policy, which can be interpreted as fuzzy rules in fuzzy logic control, and create a new action by combining the activated actions from fuzzified states. To demonstrate the effectiveness of the proposed algorithm, learning property is investigated in omni-directional mobile robot control through simulation.

To demonstrate the effectiveness of the proposed algorithm, learning property is investigated in omni-directional mobile robot control to generate smooth trajectory through simulation. The simulation result confirms that the performance of proposed algorithm is better than the conventional Q-learning method.

This paper is organized as follows: Section II describes brief review on Q-learning. Section III proposes an algorithm, which uses fuzzified states and weighted actions to generate a sequence of continuous valued actions. Section IV illustrates the proposed algorithms on simulation examples of the omni-directional mobile robot. Finally, concluding remarks follow in Section V.

## II. PRELIMINARIES

In reinforcement learning, the purpose of agent is to discover the optimal action in an environmental state so as to maximize a reward in long term. Q-learning, one of the well-known schemes in the reinforcement learning, is easy to implement and is not affected by a learning policy. The value of taking action, $a$, in state, $s$, under a policy, $\pi$, is called a state-action value and denoted as $Q^{\pi}(s, a)$. Simply, this value represents a value for each action from each state, which depends on a received reward, current value and other values as well. As the state-action value, $Q$, converges to the optimal value, the agent selects an action with the maximum $Q$ value in a given state.

In Q-learning, agent has the ability to learn state-action value using selected actions in the state without having the model of environment. Updating equation of the $Q$ value is as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)) \tag{1}$$

where $\alpha$ is the learning rate and $\gamma$ is the discount rate. As $\gamma$ gets closer to 0, the agent will tend to consider only

Authors are with the Department of Electrical Engineering, KAIST, Guseong-dong, Yuseong-gu, Daejeon, 305-701, Republic of Korea {dhlee, iwpark, johkim}@rit.kaist.ac.kr

TABLE I

PSEUDO CODE FOR Q-LEARNING

Initialize all $Q(s, a)$ values
**Repeat** (for each episode)
  Choose a starting state, $s$
  **Repeat** (for each step)
    Choose an action, $a$, at s using policy derived from $Q$
    Take the action, $a$, and observe a reward, $r$, and a next state, $s'$
    $Q(s, a) \leftarrow Q(s, a)$ + $\alpha$( $r$ + $\gamma$ $\max_{a'}$ $Q(s', a')$ - $Q(s, a)$)
    $s \leftarrow s'$
  **Until** $s'$ is a goal state
**Until** a desired number of episodes terminated

immediate reward, whereas the agent will consider future reward with greater weight as $\gamma$ becomes closer to 1.

Table I shows the pseudo code for Q-learning algorithm. Once an action is selected, the agent receives an immediate reward and $Q$ value of state-action pair is updated. Since excessive exploratory action selections prevent the progress of Q-learning, the trade-off between exploration and exploitation is needed. For instance, in the $\epsilon$-greedy method, the $\epsilon$ value decreases as the number of episode increases to be exploration in the beginning and exploitation as the learning progresses. Note that each episode ends at a goal state, followed by a reset to a starting state.

Unlike temporal-difference learning, actor-critic learning, and SARSA, Q-learning is off-policy learning. In other words, the state-action value function of a selected action made by the agent is learned irrespective of the policy due to the presence of $\max$ operator in (1). Thus, Q-learning is insensitive to the policy of exploration. Even though the trade-off between exploration and exploitation is needed in the Q-learning similarly as in other reinforcement learning algorithms, the convergence of $Q$ value is not affected by the policy of exploration. Consequently, $Q$ value of the state-action pairs is to converge to the optimal value if the state and action pairs are visited many times by the agent.

## III. Q-LEARNING USING FUZZIFIED STATES AND WEIGHTED ACTIONS

The conventional Q-learning method uses a finite number of discretized states and discretized actions to learn a certain policy. However, agent chooses only one action for one state, which is not appropriate to generate a continuous sequence of actions in continuous environment. Thus, this paper proposes a new method of Q-learning that uses fuzzified states and weighted actions.

When the fuzzified states are used, agent can be positioned in several states at the same time and generates a new action by combining the actions corresponding to the number of states. A dominant state, $s_D$, dominant action, $a_D$ and state weight, $w_i$, are defined to specify an action. The dominant state is the state, which has the largest membership value among other states. The dominant action is the action, which maximizes the $Q$ value at the dominant state. The state weight is defined as the normalized membership value, which can be used as a parameter of an action, such as the magnitude of the action.

TABLE II

PSEUDO CODE FOR Q-LEARNING USING FUZZIFIED STATES AND WEIGHTED ACTIONS

Initialize all $Q(s, a)$ values
Define fuzzy set
**Repeat** (for each episode)
  Choose a starting state, $s$
  **Repeat** (for each step)
    Find out all fuzzified states ($n$) using the fuzzy set
    Calculate state weights
    Find out the dominant state, $s_D$, which has the largest weight, $w$
    Choose the dominant action, $a_D$, at $s_D$ using policy derived from $Q$
    Actions of other fuzzified states, $a_i$ = max $Q(s_i, a)$
    Final action, $a_F$ = $\sum_{i=1}^{n}$ $w_i \cdot a_i$
    Take the action, $a_F$, and observe a reward, $r$, and a next state, $s'$
    $Q(s_D, a_D) \leftarrow Q(s_D, a_D)$
      + $\alpha$( $r$ + $\gamma$ $\max_{a'}$ $Q(s', a')$ - $Q(s_D, a_D)$)
    $s \leftarrow s'$
  **Until** $s'$ is a goal state
**Until** a desired number of episodes terminated

The pseudo-code of the proposed Q-learning using fuzzified states and weighted actions is shown in Table II. Firstly, find all states where the robot belongs and calculates the state weights of all states. Secondly, define the dominant state, $s_D$, which has the largest state weight. Then it finds the dominant action, $a_D$, which maximizes $Q$ value at $s_D$ by using the policy. Thirdly, calculate the final action, $a_F$, by adding the multiples of the state weight and each action corresponds to each state. Finally, update the $Q$ value corresponds to $s_D$ and $a_D$ pair. The action of dominant state is selected by the policy derived from the state-action value, $Q$, such as $\epsilon$-greedy method or softmax action selection method. Note that all actions of other fuzzified states are selected by the greedy method. As shown in Table II, only the state-action value of dominant state gets updated after receiving an immediate reward.

As an example, assume that the agent is located in three fuzzified states, $s_1$, $s_2$ and $s_3$. The corresponding membership values are $\mu_{s_1}(x) = 0.5$, $\mu_{s_2}(x) = 0.3$ and $\mu_{s_3}(x) = 0.1$, respectively, and the corresponding actions to the states are $a_1$, $a_2$ and $a_3$, respectively. Since the largest membership value is $\mu_{s_1}(x)$, $s_1$ becomes the dominant state, $s_D$, and $a_1$ becomes the dominant action, $a_D$. The rates based on the the membership value of the dominant state become $1 : 0.6 : 0.2$. By normalizing these rates, the state weights for the states are calculated as $w_1 = \frac{1}{1.8}, w_2 = \frac{0.6}{1.8}$ and $w_3 = \frac{0.2}{1.8}$, respectively. If the action is defined as a vector, which represents the magnitude and direction of agent behavior, each of the state weight is multiplied to the action and all of these weighted actions are lastly added up to generate a final action, $a_F$, as follows:

$$a_F = w_1 \cdot a_1 + w_2 \cdot a_2 + w_3 \cdot a_3 \tag{2}$$

As a result, $a_F$ is executed by the robot and the $Q$ value of $s_D$ and $a_D$ pair, $Q(s_1, a_1)$ is updated from (1).

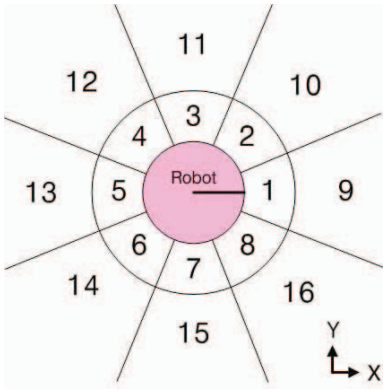



Fig. 1. State representation of omni-directional mobile robot

## IV. COMPUTER SIMULATION

In the simulation, an omni-directional mobile robot was trained to reach a target with the minimum elapsed time. During the training, the target was placed to one state until the robot could reach the target and then the target was moved to the next state. This procedure was iterated until the robot could reach the target which was placed in any state. During the test, the elapsed time of the robot to reach all distributed targets in order was measured. The conventional Q-learning method and the proposed method were applied to train the robot and the elapsed time was used to compare the performances of two methods.

The states of learning system were defined according to the target position from robot as shown in Fig. 1. Each of the states was defined as a fuzzy set of distance, $d$, and angle, $\theta$, to the target as shown in Fig. 2. Fig. 2(a) represents the fuzzy set of distance to the target, where the radius of robot is represented as, $r$. Fig 2(b) represents the fuzzy set of angle to the target. By using two fuzzy sets, the fuzzified states were obtained as shown in Table III. Consequently, robot considered four fuzzified states from the distance and angle to the target. By using the membership values of the fuzzified states, the corresponding state weights could be calculated.

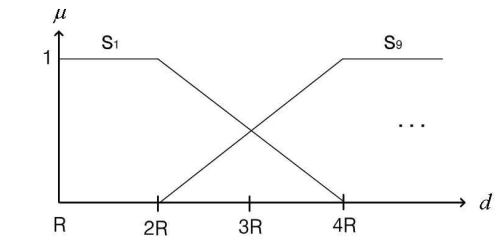
There were six actions of mobile robot: moving forward (FWD), backward (BWD), left (LEFT), right (RIGHT), rotating clockwise (CW) and counter-clockwise (CCW). Each of the actions was defined as a vector of the desired step distance as shown in Table IV.

TABLE III
FUZZIFIED STATES FROM FUZZY SETS

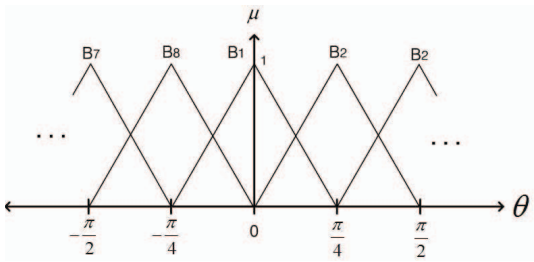
| | $B_1$ | $B_2$ | $B_3$ | $B_4$ | $B_5$ | $B_6$ | $B_7$ | $B_8$ |
|---|---|---|---|---|---|---|---|---|
| $D_1$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ | $S_6$ | $S_7$ | $S_8$ |
| $D_2$ | $S_9$ | $S_{10}$ | $S_{11}$ | $S_{12}$ | $S_{13}$ | $S_{14}$ | $S_{15}$ | $S_{16}$ |

TABLE IV
ACTION VECTORS

| | FWD | BWD | LEFT | RIGHT | CW | CCW |
|---|---|---|---|---|---|---|
| $x\ (m)$ | 1 | -1 | 0 | 0 | 0 | 0 |
| $y\ (m)$ | 0 | 0 | 1 | -1 | 0 | 0 |
| $\theta\ (rad)$ | 0 | 0 | 0 | 0 | $-\frac{\pi}{4}$ | $\frac{\pi}{4}$ |




(a)




(b)

Fig. 2. Fuzzy sets of learning system (a) distance to the target (b) angle to the target

The reward function was defined as follows:

$$rwd(d,\theta) = -\alpha \cdot min(d, D_{max}) - \beta \cdot |\theta| \qquad (3)$$

where $D_{max}$ equals $10 \cdot r$, and $\alpha$ and $\beta$ are constants. The robot received a higher reward as it tried to minimize the distance and angle from the target.

The $\epsilon$-greedy method was applied to both learning methods. It chose an action, which has the largest $Q$ value at a state, with the probability of $1-\epsilon$ and the other actions had the probability of $\epsilon$ to be chosen. By using the $\epsilon$-greedy method, we could generate different policies. We made five policies from each of the methods.

Table V shows the five trained policies from the conventional Q-learning method, where Table VI represents the five policies generated from the proposed Q-learning using fuzzified states and weighted actions. Note that when the policies in Table V were applied to the robot, the discretized states were used so that the robot could locate in only one state at a time and only one action which corresponds to the state was executed. When the policies in Table VI were used, on the other hand, the fuzzified states were used so that the robot could locate in more than one state at a time and all of the actions which correspond to the states were multiplied by the state weights and then added to generate the final action.

The policy which was trained from the conventional Q-learning method and the policy from the proposed method were represented as $Pol^C$ and $Pol^F$, respectively. To test the generated policies, seven different targets were used for

TABLE V

RESULTS OF POLICIES FROM CONVENTIONAL Q-LEARNING

| | $Pol_1^C$ | $Pol_2^C$ | $Pol_3^C$ | $Pol_4^C$ | $Pol_5^C$ |
|---|---|---|---|---|---|
| $S_1$ | FIN | FIN | FIN | FIN | FIN |
| $S_2$ | CCW | CCW | CCW | CCW | CCW |
| $S_3$ | CCW | CCW | CCW | CCW | CCW |
| $S_4$ | CCW | CCW | CCW | CCW | CCW |
| $S_5$ | CW | CW | CW | CW | CCW |
| $S_6$ | CW | CW | CW | CW | BWD |
| $S_7$ | CW | CW | CW | CW | CW |
| $S_8$ | CW | RIGHT | CW | RIGHT | CW |
| $S_9$ | FWD | FWD | FWD | FWD | FWD |
| $S_{10}$ | LEFT | CCW | CCW | CCW | LEFT |
| $S_{11}$ | LEFT | CCW | LEFT | LEFT | LEFT |
| $S_{12}$ | CCW | CCW | CCW | CCW | CCW |
| $S_{13}$ | BWD | BWD | BWD | BWD | BWD |
| $S_{14}$ | CW | CW | CW | CW | CW |
| $S_{15}$ | RIGHT | CW | RIGHT | CW | CW |
| $S_{16}$ | CW | CW | RIGHT | CW | CW |

TABLE VI

RESULTS OF POLICIES FROM Q-LEARNING USING FUZZIFIED STATES

| | $Pol_1^F$ | $Pol_2^F$ | $Pol_3^F$ | $Pol_4^F$ | $Pol_5^F$ |
|---|---|---|---|---|---|
| $S_1$ | FIN | FIN | FIN | FIN | FIN |
| $S_2$ | RIGHT | FWD | CCW | LEFT | CCW |
| $S_3$ | RIGHT | LEFT | CCW | CCW | CCW |
| $S_4$ | BWD | LEFT | CCW | BWD | CCW |
| $S_5$ | FWD | CW | FWD | CCW | BWD |
| $S_6$ | CW | CW | CW | RIGHT | BWD |
| $S_7$ | CW | CW | CW | BWD | CW |
| $S_8$ | RIGHT | CW | CW | CW | RIGHT |
| $S_9$ | FWD | FWD | FWD | FWD | FWD |
| $S_{10}$ | CCW | CCW | CCW | CCW | CCW |
| $S_{11}$ | CCW | CCW | LEFT | CCW | LEFT |
| $S_{12}$ | CCW | CCW | CCW | CCW | CCW |
| $S_{13}$ | CCW | LEFT | CW | CCW | CCW |
| $S_{14}$ | CCW | BWD | CW | FWD | CCW |
| $S_{15}$ | RIGHT | CW | RIGHT | CW | FWD |
| $S_{16}$ | CW | RIGHT | CW | RIGHT | CW |

the test set. Only one target appeared at a time and as soon as the robot reached to the target, the current target was disappeared and the next target was generated. Four different test sets were used to test $Pol^C$ and $Pol^F$ and the elapsed time of the robot finishing the test was measured.

Table VII and Table VIII show the results of elapsed time by using the policy $Pol^C$ and $Pol^F$, respectively. As shown in Table VII and Table VIII, the elapsed times with using the fuzzified policy, $Pol^F$, was faster than those of using the conventional policy, $Pol^C$. Since the speed of the robot was fixed, it could be inferred that the trajectory generated by the $Pol^F$ must be more efficient than that of $Pol^C$ to obtain the lower elapsed time.

To verify the inference, the trajectories of robot by using $Pol^C$ and $Pol^F$ are shown in Fig. 3 and Fig. 4, respectively (refer to the last page). As shown in Fig. 3, the trajectory of robot using the general policy, $Pol^C$, showed that robot abruptly changed its direction and moved in an inefficient way. For example, in Fig. 3(a), when the robot tried to reach $Target_1$, it firstly moved left and then moved forward. On the other hand, the trajectory using the fuzzified policy, $Pol^F$, showed that robot was able to change its direction

TABLE VII

ELAPSED TIME WHEN CONVENTIONAL POLICY IS USED

| | $Pol_1^C$ | $Pol_2^C$ | $Pol_3^C$ | $Pol_4^C$ | $Pol_5^C$ | Average |
|---|---|---|---|---|---|---|
| $Test_1\ (sec)$ | 87.7 | 73.5 | 85.1 | 76.0 | 87.7 | 82.0 |
| $Test_2\ (sec)$ | 92.3 | 92.4 | 101.4 | 90.3 | 92.3 | 93.7 |
| $Test_3\ (sec)$ | 90.6 | 93.2 | 92.3 | 91.5 | 95.6 | 92.6 |
| $Test_4\ (sec)$ | 96.8 | 96.7 | 100.5 | 91.6 | 98.0 | 96.7 |

TABLE VIII

ELAPSED TIME WHEN FUZZIFIED POLICY IS USED

| | $Pol_1^F$ | $Pol_2^F$ | $Pol_3^F$ | $Pol_4^F$ | $Pol_5^F$ | Average |
|---|---|---|---|---|---|---|
| $Test_1\ (sec)$ | 70.5 | 67.5 | 62.9 | 71.7 | 62.9 | 67.1 |
| $Test_2\ (sec)$ | 82.8 | 77.8 | 75.4 | 83.2 | 75.7 | 78.9 |
| $Test_3\ (sec)$ | 82.9 | 74.8 | 82.1 | 75.8 | 81.4 | 79.4 |
| $Test_4\ (sec)$ | 85.1 | 73.7 | 84.9 | 75.5 | 86.6 | 81.1 |

smoothly and move in more efficient way compared to the $Pol^C$. For example, in Fig. 4(a), when the robot tried to reach $Target_1$, it could change its direction and reach to the target continuously by combining all of the actions which correspond to the states where the robot was located in.

## V. CONCLUSIONS

We proposed the Q-learning algorithm, which uses fuzzified states and weighted actions. By changing the discretized states of conventional Q-learning to the fuzzified states, the learning system was able to cope with the changes of states in continuous domain, where the state weights represented the weights of states. By multiplying the state weights to discretized actions and then adding all the weighted actions correspond to the fuzzified state, the final action that correctly responded to the states in continuous domain was generated.

The proposed algorithm was applied to the target tracking of the omni-directional mobile robot control. As a result, the trajectory of the omni-directional mobile robot was smoother and more efficient than the conventional Q-learning approach when the proposed algorithm was applied.

## VI. ACKNOWLEDGMENTS

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2009-0080432)

## REFERENCES

[1] P. Y. Glorennec, "Fuzzy Q-Learning and Dynamical Fuzzy Q-Learning," *Proc. of IEEE Int. Conf. on Fuzzy Systems*, pp. 474-479, Jun. 1994.

[2] P. Y. Glorennec and L. Jouffe, "Fuzzy Q-learing," *Proc. of IEEE Int. Conf. on Fuzzy Systems*, pp. 659-662, Jul. 1997.

[3] H. R. Berenji, "Fuzzy Q-learning: A New Approach for Fuzzy Dynamic Programming," *Proc. of IEEE Int. Conf. on Fuzzy Systems*, pp. 486-491, Jun. 1994.

[4] H. R. Berenji, "Fuzzy Q-learning for Generalization of Reinforcement Learning," *Proc. of IEEE Int. Conf. on Fuzzy Systems*, pp. 2208-2214, Sep. 1996.

[5] L. Busoniu, D. Ernst, B. D. Schutter and R. Babuska, "Fuzzy Approximation for Convergent Model-based Reinforcment Learning," *Proc. of IEEE Int. Conf. on Fuzzy Systems*, pp. 1-6, Jul. 2007.

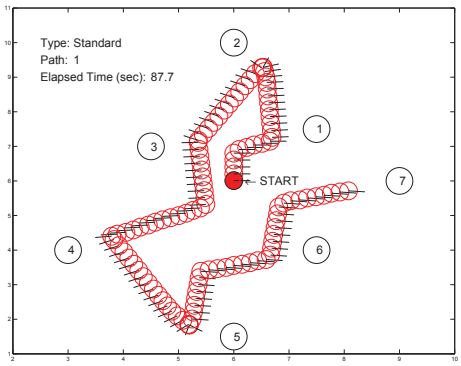
[6] L. Busoniu, D. Ernst, B. D. Schutter and R. Babuska, "Continuous-State Reinforcement Learning with Fuzzy Approximation," *Lecture Notes in Computer Science*, vol. 4865, pp. 27-43, 2008

[7] H. R. Berenji and D. Vengerov, "A Convergent Actor-critic-based FRL Algorithm with Application to Power Management of Wireless Transmitters," *IEEE Transactions on Fuzzy Systems*, vol. 11, no. 4, pp. 478-485, 2003.

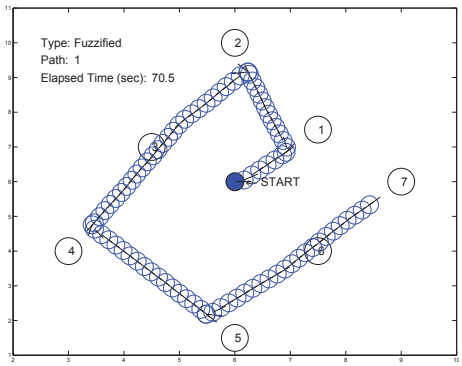
[8] D. Vengerov, N. Bambos and H. R. Berenji, "A Fuzzy Reinforcement Learning Approach to Power Control in Wireless Transmitters," *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 35, no. 4, pp. 768-778, 2005.

[9] Y. Duan and X. Hexu, "Fuzzy Reinforcement Learning and Its Application in Robot Navigation," *Proc. of Int. Conf. on Machine Learning and Cybernetics*, pp. 899-904, Aug. 2005.

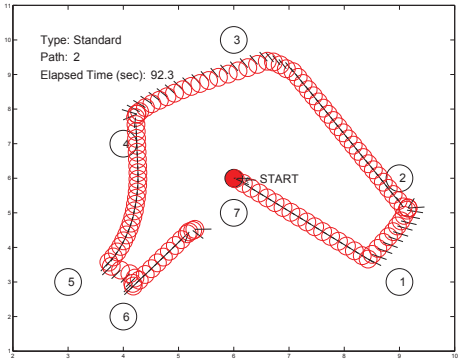
[10] Z. Zhai, W. Chen, X. Li, and J. Guo, "A Modified Average Reward Reinforcement Learning based on Fuzzy Reward Function," *Proc. of Int. Conf. Engineers and Computer Scientists*, pp. 113-117, 2009.



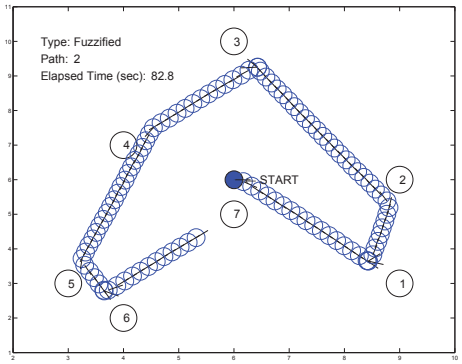

(a)



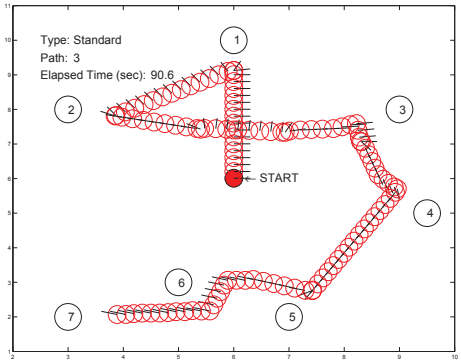

(a)



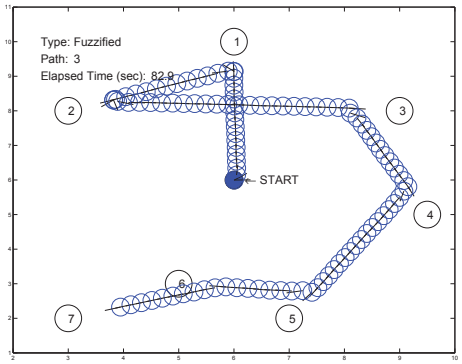

(b)



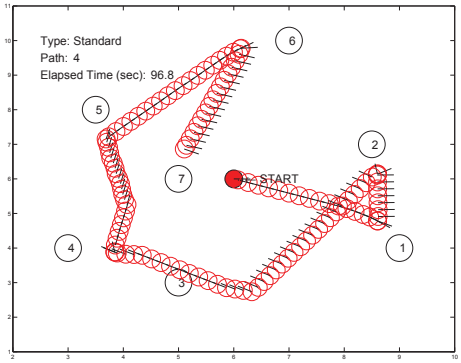

(b)



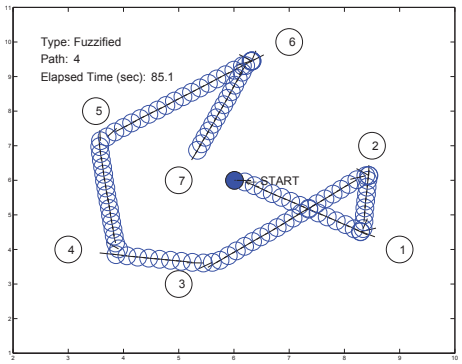

(c)




(c)




(d)




(d)

Fig. 3. Trajectories of omni-directional mobile robot by using $Pol_1^C$

Fig. 4. Trajectories of omni-directional mobile robot by using $Pol_1^F$