Region-based Q-Learning for Intelligent Robot Systems

I.H.Suh[1], J.H.Kim[1], and S.R.Oh[2]
[2] Intelligent Control and Robotics Lab. Dept. of Electronics Eng., Hanyang Univ.,
396 Daehak-dong Ansan-Si, Kyeongki-Do 425-791, Korea
[3] Advanced Robotics Research Center, KIST, Seoul, Korea
Fax no : +82-345-408-5803
E-mail : ihsuh@shira.hanyang.ac.kr

Abstract

It is desirable for autonomous robot systems to possess the ability to behave in a smooth and continuous fashion when interacting with an unknown environment. Since Q-learning requires a lot of memory and time to optimize a series of actions in a continuous state space, it may not be easy to apply the method to such a real environment. In this paper, for a continuous state space applications, we propose a new method of Q-learning that incorporates a region-based reward assignment being used to solve structural credit assignment problem and a triangular type Q-value model. Our learning method can estimate a current Q-value by a relationship with its neighboring states and has the ability to learn its actions similar to that of Q-learning. Thus, our method can enable robots to move smoothly in a real environment. To show the validity of our method, navigation comparison with Q-learning are given and visual tracking simulation results involving an 2-DOF SCARA robot are also presented

Keywords : region-based reward assignment, Q-value model , neighboring states, visual tracking

1. Introduction

Recently, many intelligent robot systems have been developed to perform various tasks which include navigation, control, and recognition [1~3]. To implement such a robot system, it is important for the system to properly react in an unknown environment by learning its actions through experience. For this purpose, reinforcement learning methods have been receiving increased attention for use in robot systems due to its appealing attributes.

Prior works on reinforcement learning have been conducted on the basis of discrete states and events [4~6]. By using such learning methods, a robot can learn its proper actions by interacting with an unknown environment. Among these methods, Q-learning has been widely used, where Q-values to be generated can be considered as action evaluation values. These values are then used to find the proper action in each state. However, since Q-learning deals with discrete actions and states, an enormous amount of Q-value information may be necessary for an autonomous robot to learn an appropriate action in a continuous environment. Hence, this may require a great deal of time and memory to learn the proper actions for all the states. To overcome this problem, variations of the Q-learning algorithm have been developed [7~9]. In one of these algorithms, the Q-values are evaluated by using Tagaki-Sugeno-Kang (TSK) type fuzzy rules [9], in which a set of consequent coefficients are modified by a steepest descent method. Since the Q-values generated by the rule base does not posses statistical information, it cannot ensure that the algorithm will find the optimal policy [9].

To solve such a problem, we propose a new method of Q-learning that deals with continuous actions and states. In reinforcement learning, one of the well-known important issues is to resolve a credit assignment problem, that is how optimal actions could be learned from the sensor-action-feedback training sequences. The structural credit assignment problem can be given as figuring out the distribution of rewards across the state space. In this viewpoint, the previous Q-learning method may be recognized as a point-based credit assignment method. Therefore, we suggest a region-based credit assignment approach to generalize the conventional Q-learning. In our method, the reward of current state propagates to its predefined *neighboring states*. From the propagated reward, Q-values and actions for *neighboring states* are updated. Then, the Q-values and actions are used to determine Q-value and action for the current action. For this, we propose a Q-value distribution model that can be

used to rapidly compute a current maximum Q-value. Based on this model, a Q-value is updated on every reinforcement signal using the well-known Q-value update equation. It is also shown that the action for any neighboring state, if it is determined by our region-based Q-learning approach, converges to the optimal action as in a pointwise Q-learning technique.

2. Q-learning in Discrete state space

Reinforcement type learning methods can be modeled as a discrete time cyclic interaction between an autonomous robot and an environment[6]. The interaction process can be explained as follows. First, the robot senses the current state and executes an action for that state. Then, a change in the environment creates the next state and assigns the robot a reward for that particular action. Based on the reward, the robot evaluates the exerted action in that state. Q-Learning provides a robot (agent) with the ability to act optimally by evaluating the consequences of its actions. Specifically, a Q-value evaluate function is used to estimate the future reinforcement for performing its actions. This enables a robot to select a proper action for a particular state. The Q-learning algorithm can be summarized as follows.

Q-learning Algorithm

<Initialize>

1. Initialize the Q-table by assigning arbitrary values.
2. Construct an initial policy f_i based on the initial values in the Q-table. That is,

$$f_i \leftarrow a \text{ such that } Q_i^a(t+1) = \max_{b \in A}\{Q_i^b(t)\}, \qquad (1)$$

where t is the tth iteration, i is the current state, f_i is the action of the current state in the policy table, A is a set of available actions, and $Q_i^a(t+1)$ is the Q-value of current action a in current state i in the next iteration.

<Repeat forever>

3. For the current state i, execute an action a based on the policy table with probability p or a random action with probability 1-p. The random action plays an important role in obtaining the optimal policy.
4. Receive a reward r from the environment.
5. Update the Q-value using

$$Q_i^a(t+1) = \alpha Q_i^a(t) + (1-\alpha)(r_i^a + \gamma \max_{b \in A}\{Q_{i+1}^b(t)\}), \qquad (2)$$

where α ($0 < \alpha < 1$) can be considered as a learning rate and γ as a discounting factor.

6. Update the policy f.

It is to be noted that the Q-value for an optimal action can be defined by combining the immediate reward r, transition probability function T, and the Q-value for the next state as

$$Q_i^a(t+1) = r(t) + \gamma \sum_{i+1 \in S} T(i,a,i+1) \max_{b \in A}\{Q_{i+1}^b(t)\}, \qquad (3)$$

if transition probability function T is given *a priori*.

Now, we will show that the Q-value update equation in (2) is equivalent to (3) as the number of iterations approaches to infinity. For this, note that $Q_i^a(t+1) = Q_i^a(t)$ in the steady state in Q-learning. Thus, we can obtain

$$Q_i^a(t) = r_i^a(t) + \gamma \max_{b \in A}\{Q_{i+1}^b(t)\}, \qquad (4)$$

if we let $T(i,a,i+1)=1$ for all i and a. Hence, (3) becomes identical to (4).

3. Region-based Q-Learning(RQ-learning)

The conventional Q-learning in Sec.2 requires too much memory and time for a robot to learn optimal actions for all states in real environment. Furthermore, since it generate discrete actions in discontinuous state space, the robot cannot perform a smooth motion. To solve the problem, we generalize the conventional Q-learning by using a region-based credit assignment technique. The region-based Q-learning may be considered as an extension of pointwise Q-learning that was mentioned in Section 2. Unlike the conventional Q-learning method, RQ-learning needs not to learn every state in the state space. In fact, optimizing the actions for only a few representative states is needed. With this in mind, we propose a reward assignment technique to generate Q-values for neighboring states. Then, we will determine an action for a current state by maximizing a linear combination of optimal Q-values for neighboring states associated with the current state, where the Q-value model of the triangular type is employed.

3.1 Determination of Q-values for neighboring states

Let each axis in N-Dimensional space have l resolution. Then, there are N^l number of contiguous hyperboxes in the space. In this state configuration, we will define a *neighboring state* as a state located in each vertex of hyperboxes surrounding current state as shown in Fig. 1. For a current reward to affect its neighboring states, we will propose an *effect function*, $\mu_{i,j}(s_i,s_{i,j})$, that relates r_i, the reward of current state s_i which can be located everywhere inside a hyperbox, with r_j, the rewards of neighboring states $s_{i,j}$ which are vertices of the hyperbox including s_i. By using *effect function*, we will obtain rewards for neighboring states given as

$$r_j = \mu_{i,j} r_i. \qquad (5)$$

As shown in Fig.1, a reward of a neighboring state can be obtained by multiplying $\mu_{i,j}(s_i,s_j)$ and r_j. Thus, the

expected sum of rewards propagated to s_{ij} can be written as

$$Q_j^{a_i} = \sum_{n=0}^{x} \gamma^n \mu_{i+n,j} r_{i+n} \tag{6}$$

Theorem 1

The action maximizing $Q_j^{a_i}$ in Eq. (6) converges to the optimal action, as iteration number increases, if $r_j = u_{i,j} r_i$ as given in Eq.(5).

Proof

$$Q_j^{a_i}(t) = \sum_{n=0}^{x} \gamma^n u_{i+n,j} r_{i+n} . \tag{7}$$

$$= \mu_{i,j} r_i + \sum_{n=1}^{\infty} \gamma^n \mu_{i+n,j} r_{i+n} . \tag{8}$$

$$= \mu_{i,j} r_i + \gamma \sum_{n=1}^{\infty} \gamma^{n-1} \mu_{i+n,j} r_{i+n} . \tag{9}$$

$$= \mu_{i,j} r_i + \gamma \max_{b \in A} \{Q_j^b(t)\} . \tag{10}$$

When $Q_j^{a_i}(t+1)$ is made to be updated as in conventional Q-learning equation in Eq.(2), we obtain that

$$Q_j^{a_i}(t+1) = \alpha Q_j^{a_i}(t) + (1-\alpha)\left[\mu_{i,j} r_i + \gamma \max_{b \in A} \{Q_j^b(t)\}\right] \tag{11}$$

Since $r_j = \mu_{i,j} r_i$ as in Eq.(5), $Q_j^{a_i}(t+1)$ can be rewritten as

$$Q_j^{a_i}(t+1) = \alpha Q_j^{a_i}(t) + (1-\alpha)\left[r_j + \gamma \max_{b \in A} \{Q_j^b(t)\}\right] . \tag{12}$$

It is here noted that Eq.(12) is completely equivalent to Eq.(2), and the action maximizing Q-value given by Eq.(2) has been proved to converge to the optimal one by Watkins[5]. Thus, the action maximizing $Q_j^{a_i}$, Q-value of neighboring states converges to the optimal action, when r_j is estimated as $\mu_{i,j} r_i$. This completes the proof.□

It is remarked that if all vertices of each hyperbox are concentrated to the center of the hyperbox, we can have N^l number of l-equidistanced discrete pointwise hyperboxes. In this case, Euclidean distance between s_{ij} and $s_{i,j+1}$, $d(s_{i,j}, s_{i,j+1})$, for a hyperbox should be null due to null volume of the hyperbox. This implies that a reward owing to a current state should be fully given to all neighboring states. After all, Q-values for all discrete states can be obtained by conventional Q-value update equation in Eq.(2). Therefore, for $Q_j^{a_i}$ in Eq.(6) to be a generalization of the conventional Q-learning, following property should hold for $\mu(s_i, s_{i,j})$;

$$\lim_{d(s_i, s_j) \to 0} \mu_{i,j} \to 1 \tag{13}$$

Note that it will be difficult to obtain N^l number of *effect functions* to be appropriate for each hyperbox. Therefore, we will use the *effect function* $\mu_{i,j}(s_i, s_j)$ of the same type for all hyperboxes. For the sake of simplicity, we choose the *effect function* given by

$$\mu_{i,j}(s_i, s_{i,j}) = \exp(-d(s_i, s_{i,j})) , \tag{14}$$

where $d(x,y)$ is the Euclidean distance between the states x and y. It is easily observed that Eq.(14) satisfy the property in Eq.(13).

3.1 Q-table model of the triangular type for a continuous state space

For a Q-learning scheme to perform satisfactorily in a continuous state space, an infinite number of states and actions in the Q-table is required. Hence, to remedy this constraint, we use a continuous Q-table model of the triangular type in which a particular action has the highest Q-value and the opposite action has a zero value. Thus, the relationship among all the actions is simply defined as an Euclidean distance. This is shown in Fig. 2, where Q_{max} and a_{max} denote the maximum amplitude and width, respectively.

3.2. Action generation in Region-based Q-learning

Using the above definition, the action for the current state is simply determined as the maximum of all Q-values. That is,

$$a_i = \arg(\max_{\forall a} \{\sum_{j=1}^{\lambda} \mu_{i,j} Q_{i,j}^a\}) . \tag{15}$$

Fig. 4 shows an example of the current action generated by equation (15) for the current state from Fig. 3. After executing the current action, the current state changes to its next state and is assigned a reward. Based on the current action and the updated amount of the Q-value, we can modify the proper action to have a maximum Q-value. There exist two types of updating methods. One is, if the updated Q-value is smaller than the largest Q-value in current state and is larger than the Q-value of current action, then the largest Q-value moves in the direction of the current action to update the Q-value. In addition, if the updated Q-value is smaller than Q-value of current action, then the largest Q-value moves in the opposite direction of the current action. This update scheme can be represented by

$$a_{i,j}^k(t+1) = a_{i,j}^k(t) + \eta \, \mathrm{sgn}(a_{i,j}^k(t) - a_i^k(t)) , \tag{16}$$

where $\eta = \frac{2a_{max}^k}{Q_{max}} \left| dQ_{i,j}^{a_i}(t+1) \right|$,

In (16), η is used to determine the amount to update the k-th axis action value and sgn() is used to determine the

direction of the actions movement. Second, if the updated Q-value is larger than the current maximum Q-value, then the action is used as the one with the largest Q-value. That is,

$$a^k_{i,j}(t+1) = a^k_i(t). \quad (17)$$

As an example, Fig. 5 shows that the triangle-shaped Q-table model moves towards the current action. The fuzzy Q-learning algorithm can be summarized as follows.

RQ-Learning Algorithm

<Initialize>

1. Initialize all necessary parameters.

<Repeat forever>

2. For the current state, compute the relation values of the current state in the neighboring states based on Eq.(14).
3. Compute the current action a by Eq.(15).
4. Execute the current action and receive a reward r from the environment.
5. Compute the Q-values of its neighboring states by Eq.(12).
6. Update the current action in the current state by Eqs.(16) and (17) based on Q-table model.

4. Simulation Results

First, we simulated the comparison between the conventional Q-learning and RQ-learning in a 2-D virtual space. The robot starts from an initial feature vector (50,50) and its goal is to reach target feature vector (320,320). The resolution of each state axis is set at 35. As shown in Fig. 6, in the initial iteration, the robot wanders here and there updating the Q-values for its performed actions. However, the maximum Q-value of each state converges after 400 iterations, thus the agent follows a policy with the sum of Q-values that is maximized. On the other hand, in the case of RQ-learning, the robot behaves well under such an optimal policy after 47 iterations. This is shown in Figure 7. As expected, our RQ-learning method shows a faster convergence than the conventional Q-learning method in which the goal is achieved in a smaller number of steps. The robot can also learn a smoother action set, thus enabling it to react to a current state with a better tuned action. We also simulated the same task with a smaller resolution of each state axis (see Figure 8). We find that our RQ-learning works satisfactory as long as the state space of a rectangle region can be approximated with effect function.

Furthermore, we have also simulated a visual tracking task of a two-link robot arm. For this, we utilized a 4-dimensional feature space, where there are two joint angles of robot and two velocity features of the object. And, the action space is made of two joint angular velocities. These state and action variables are shown in Table 1.

Table 1. The state and action variables for a visual tracking task.

Variable (range)	Description
θ_{l1} (-30 ~ 210 degree)	the angle of robot link1
ΔP_x (-10 ~ +10 cm)	x-directional difference of center position between camera and object.
θ_{l2} (-30 ~ 210 degree)	the angle of robot link1
ΔP_y (-10 ~ +10 cm)	y-directional difference of center position between camera and object.
a_{l1} (-5 ~ +5 degree)	velocity of link1
a_{l2} (-5 ~ +5 degree)	velocity of link2

The reward used in this simulation is defined as

$$r = \frac{(\|x_{i+1} - x_g\| - \|x_i - x_g\|)}{K}, \quad (14)$$

where x_i and x_{i+1} denote the current and next feature vectors, respectively, x_g is a target feature vector and K is the maximum increment that the robot arm can move toward the goal state. The target object moves in random. In every sampling time, the robot is controlled by actions to the object based on our proposed RQ-learning technique to move toward the object as in Fig.9. From this simulation, it is observed that the robot adaptively moves toward the object after approximately 1000 trials.

5. Conclusion

In this paper, we proposed a new method of Q-learning that uses a region-based reward assignment technique and triangular Q-value model for approximating continuous states. In order to show the validity of our method, we conducted some computer simulations in which a simple Q-learning was compared with our method. Results show that the required number of iterations needed to reach a goal using our method was much less. Also, we illustrated that our method was suitable for a visual tracking application that requires continuous states and actions in a real environment. However, in both Q-learning and Region-based Q-learning, setting the parameter of the random action ratio can be difficult. We are currently working on an automatic estimation method to obtain these parameters by detecting the extent of learning.

6. References

[1] M. A. Salichs, E. A. Puente, D. Gachet, and J. R. Pementel, "Learning behavioral control by

reinforcement for an autonomous mobile robot", *IEEE Conference on R&A*, vol.1, pp. 1436-1441, 1993.

[2] H. Berenji , P. Khedkar, "Learning and tuning fuzzy logic controllers through reinforcement," *IEEE Trans. on Neural Networks*, vol. 3, no. 5, Sept. 1992.

[3] L. J. Lin, "Programming robots using reinforcement learning and teaching," In *Proc. of the Ninth National Conference on Artificial Intelligence*, 1991.

[4] G. Tesauro, *Practical issues in temporal difference learning*. Machine Learning, 1992.

[5] C. Watkins, P. Dayan, "Q-learning, technical note," Machine Learning, Vol. 8, pp.279-292, 1992.

[6] C. Watkins, "Learning from delayed rewards," Ph.D. Thesis, University of Cambridge, England, 1989.

[7] P. Y. Glorennec, "Fuzzy Q-learning and dynamical fuzzy Q-learning," *IEEE Conference on R&A*, vol. 1, pp. 474-479, 1994.

[8] H. R. Berenji, "Fuzzy Q-learning: A new approach for fuzzy dynamic programming," *IEEE Conference on R&A*, vol. 1, pp. 486-491, 1994.

[9] T. Horiuchi, A. Fujino, O. Katai, and T. Sawaragi, "Fuzzy interpolation-based Q-learning with continuous states and actions," *IEEE Conference on Fuzzy Systems*, vol. 1, pp. 594-600, 1996.

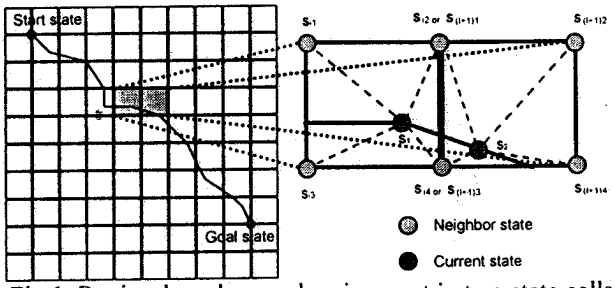


Fig.1. Region-based reward assignment in two state cells

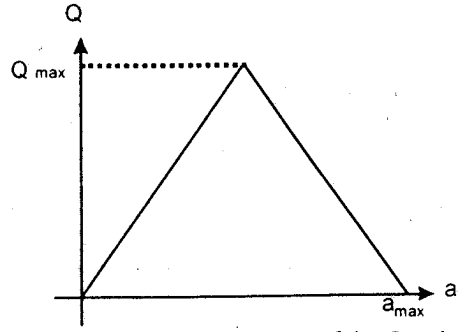


Fig 2. Triangle-type modeling of the Q-values

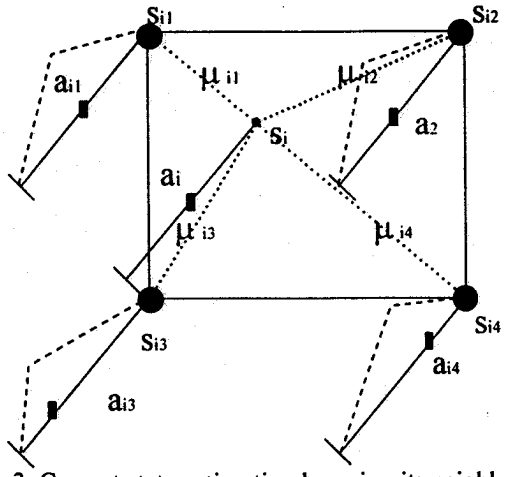


Fig. 3. Current state estimation by using its neighboring states.

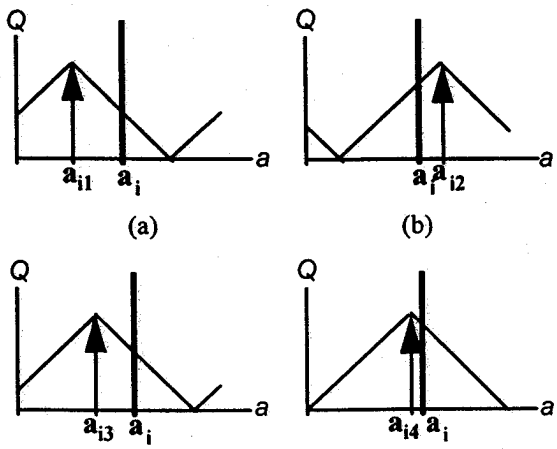
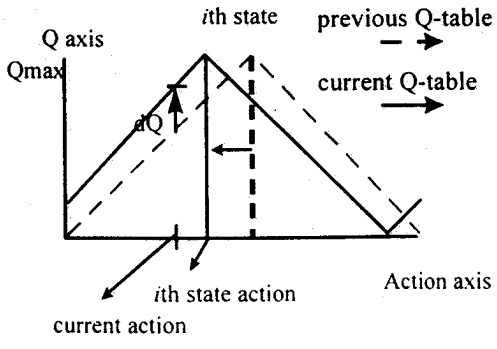
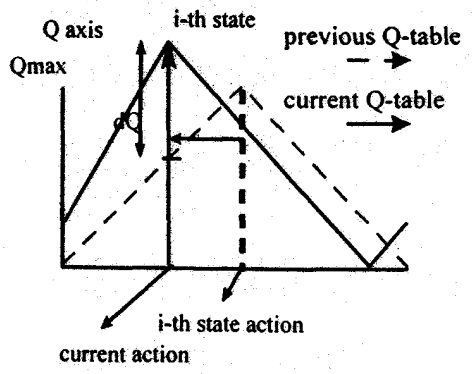


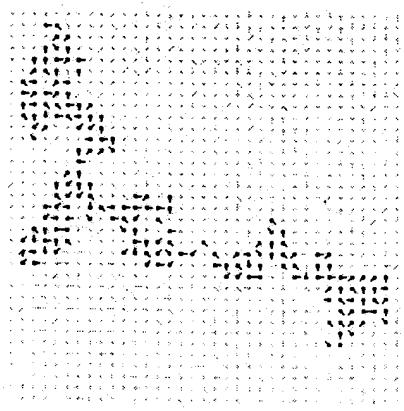
Fig. 4. An example of the current action a_i and the actions, $a_{i1}, a_{i2}, a_{i3}, a_{i4}$, of its neighboring states, $S_{i1}, S_{i2}, S_{i3}, S_{i4}$,



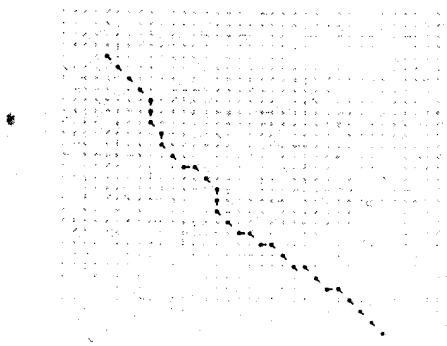


(b)

Fig. 5. The Q-value update scheme: (a) width modification and (b) height modification.

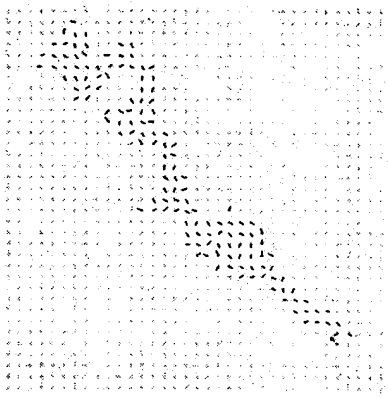


(a)

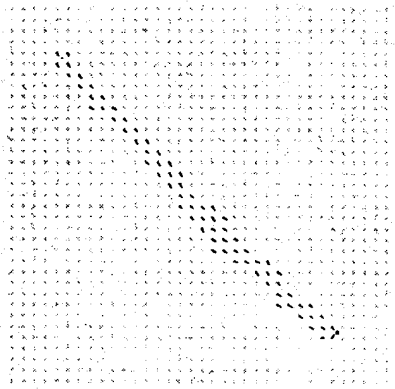


(b)

Fig. 6. The Q-learning simulation results: (a)1st iteration and (b)400th iteration.



(a)



(b)

Fig. 7. The fuzzy Q-learning simulation results: (a)1st iteration and (b)47th iteration.

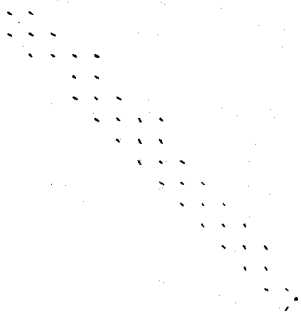
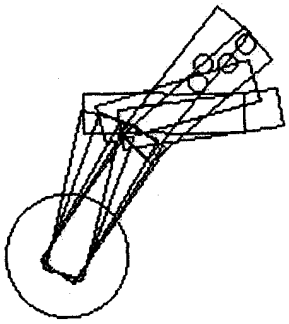
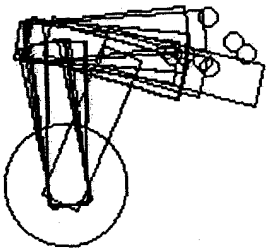


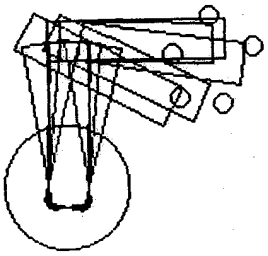
Fig. 8. The fuzzy Q-learning results using 18 resolutions for each axis.



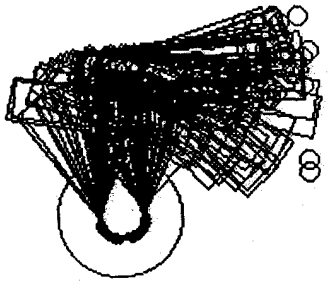
(a)



(b)



(c)



(d)

Fig. 9. Visual tracking of a 2-DOF SCARA robot by our proposed Region-based Q-learning technique.