Region-based Q-learning using Convex Clustering Approach

J.H.Kim[1], I.H.Suh[1], S.R.Oh[2], Y.J.Cho[2], and Y.K. chung[3]
[1] Intelligent Control and Robotics Lab. Dept. of Electronics Eng., Hanyang Univ.,
1271 Sa-1 dong, Ansan-Si, Kyeongki-Do 425-791, Korea
[2] Advanced Robotics Research Center, KIST, Seoul, Korea
[3] School of Mechanical Engineering, POSTECH

Abstract

In this paper*, for a continuous state space applications, a novel method of Q-learning is proposed, where the method incorporates a region-based reward assignment being used to solve structural credit assignment problem and a convex clustering approach to find a region with the same reward attribution property. Our learning method can estimate a current Q-value of an arbitrarily given state by using effect functions, and has the ability to learn its actions similar to that of Q-learning. Thus, our method enables robots to move smoothly in a real environment. To show the validity of our method, the proposed Q-learning method is compared with conventional Q-learning method through a simple two dimensional free space navigation problem, and visual tracking simulation results involving an 2-DOF SCARA robot are also presented

1 Introduction

Most of previous works on reinforcement learning have been conducted on the basis of discrete states and events [1~7]. 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 [8~10]. In one of these algorithms, Q-values are evaluated by using Tagaki-Sugeno-Kang (TSK) type fuzzy rules[10], 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 possess statistical information, it cannot ensure that the algorithm will find the optimal policy [10].

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. To generalize such a conventional Q-learning, we suggest a region-based credit assignment approach, where the region is generated by an on-line convex clustering technique. For our learning method, two important techniques will be employed. One technique is a convex clustering of state vectors proposed in[11]. The similar states are grouped into a convex state region(called a cluster), where the similarity between a given state vector and other states is measured by a specific function including rewards as an argument. That is, the similarity and reward are used for generation, contraction, and extension of a cluster. The other technique is region-based Q-learning, where the reward for a current state within a cluster propagates to its *neighboring states* that are located in vertices of the cluster. 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 state. Also, 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 cluster-based Q-learning approach, converges to the optimal action as in a pointwise Q-learning technique.

* This work has been supported in part by the KIST 2000 human robot program. All Correspondence should be addressed to Prof. I.H.Suh.(E-mail : ihsuh@shira. hanyang.ac.kr).

2 Q-learning in Discrete state space

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

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,

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

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.

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

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

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

6. Update the policy f and go to step 3.

3 RQ-learning Algorithm

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.

3.1 Generalization of conventional Q-learning by Region-based Credit Assignment.

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 surrounding the current state.

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.

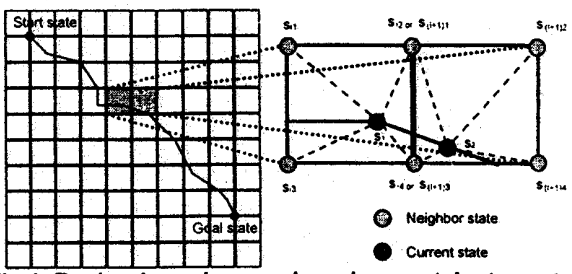


Fig.1 Region-based reward assignment in two state cells

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

$$r_j = \mu_{i,j} r_i . \quad (3)$$

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 $s_{i,j}$ can be written as

$$Q_j^{a_i} = \sum_{n=0}^{\infty} \gamma^n \mu_{i+n,j} r_{i+n} \quad (4)$$

Theorem 1

The action maximizing $Q_j^{a_i}$ in Eq. (4) converges to the optimal action, as iteration number increases, if $r_j = u_{i,j} r_i$ as given in Eq.(3).

Proof of the Theorem 1

$$Q_j^{a_i}(t) = \sum_{n=0}^{\infty} \gamma^n u_{i+n,j} r_{i+n} . \quad (5)$$

$$= \mu_{i,j} r_i + \sum_{n=1}^{\infty} \gamma^n \mu_{i+n,j} r_{i+n} . \quad (6)$$

$$= \mu_{i,j} r_i + \gamma \sum_{n=1}^{\infty} \gamma^{n-1} \mu_{i+n,j} r_{i+n} . \quad (7)$$

$$= \mu_{i,j} r_i + \gamma \max_{b \in A}\{Q_j^b(t)\} . \quad (8)$$

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

$$Q_j^{a_i}(t+1) = \alpha Q_j^{a_i}(t) + (1-\alpha)\left[\mu_{i,j} r_i + \gamma \max_{b \in A}\{Q_j^b(t)\}\right] \quad (9)$$

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

$$Q_j^{a_i}(t+1) = \alpha Q_j^{a_i}(t) + (1-\alpha)\left[r_j + \gamma \max_{b \in A}\{Q_j^b(t)\}\right]. \quad (10)$$

It is here noted that Eq.(10) 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.(4) to be a generalization of the conventional Q-learning, following property should hold for $\mu_{i,j}(s_i, s_{i,j})$;

$$\lim_{d(s_i, s_j) \to 0} \mu_{i,j} \to 1 \quad (11)$$

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* $\mu_{i,j}(s_i, s_{i,j})$ of the same type for all hyperboxes. For the sake of simplicity, we choose the *effect function* given by

$$\mu_{i,j}(s_i, s_{i,j}) = exp(-\lambda \cdot d^2(s_i, s_{i,j})), \quad (12)$$

where $d(x,y)$ is the Euclidean distance between the states x and y and λ determines the effective range. It is easily observed that Eq.(12) satisfy the property in Eq.(11).

3.2 Action generation using Q-table model of the triangular type.

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.

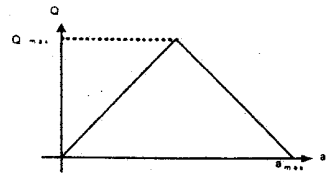


Fig 2 Triangle-type modeling of Q-values

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

$$a_i = arg(\max_{\forall a}\{\sum_{j=1}^{N} \mu_{i,j} Q_{i,j}^a\}). \quad (13)$$

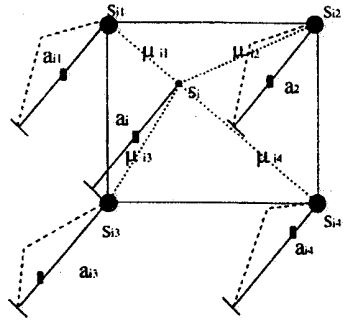


Fig. 3 Current state estimation by using its neighboring states.

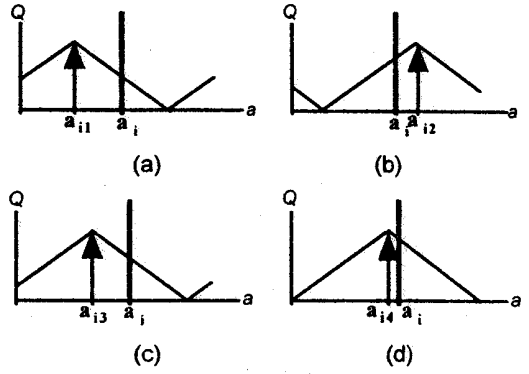


Fig. 4 An example of the current action a_i and the actions, $a_{i1}, a_{i2}, a_{i3}, a_{i4}$, of its neighboring states, $S_{i1}, S_{i2}, S_{i3}, S_{i4}$.

Fig. 4 shows an example of the current action generated by Eq. (13) 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

$$a_{i,j}^k(t+1) = a_{i,j}^k(t) + \eta\, sgn(a_{i,j}^k(t) - a_i^k(t)), \quad (14)$$

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

In Eq.(14), 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

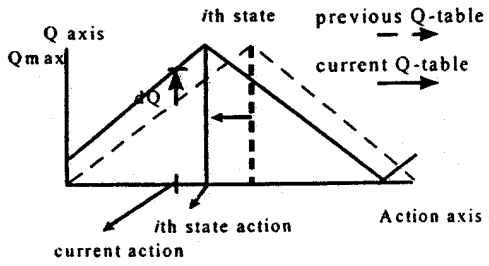
updated Q-value is larger than the current maximum Q-value, then the action is used as the one with the largest Q-value. That is,

$$a^k_{i,j}(t+1) = a^k_i(t). \quad (15)$$

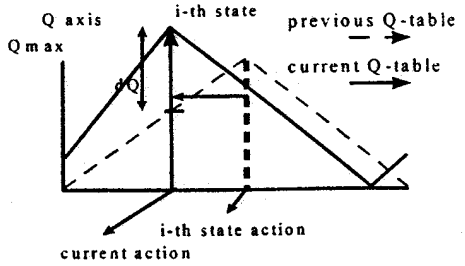
As an example, Fig. 5 shows that the triangle-shaped Q-value model moves towards the current action. The Region-based Q-learning algorithm can be summarized as follows.

RQ-Learning Algorithm

1. Initialize all necessary parameters.
2. For the current state, compute the relation values of the current state in the neighboring states based on Eq.(12).
3. Compute the current action a by Eq.(13).
4. Execute the current action and receive a reward r from the environment.
5. Compute the Q-values of its neighboring states by Eq.(10).
6. Update the current action in the current state by Eqs.(14) and (15) based on Q-table model and go to step 2.



(a)



(b)

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

4 Cluster-based Q-learning(CQ-learning)

In section 3, *effect function* has been proposed to represent the reward assignment relationship between a current state and its neighboring states. However, the reward distribution in all hyperboxes is generally difficult to be described by a predefined *effect function*. Thus, a region to be valid by an effect function may be determined by a lot of experiments. To cope with this problem, we will embed cluster-based Q-learning technique in which convex clusters having the same reward distribution property are generated by employing the method in [11].

4.1 Functional Architecture of CQ-learning

Our proposed cluster-based Q-learning system consists of mainly two functional modules, which are state clustering module and region-based Q-learning module. In state clustering module, similar states are represented as a cluster having the similar reward distribution. Based on the clusters, region-based Q-learning module proposed in section 3 performs a cluster-based Q-learning to generate a proper action for the state. Block diagram of these operations is depicted in Fig. 6.

In Fig. 6, real sensor information on environment is first passed back to the CQ-learning system. Then, necessary features are extracted and a current state is determined by such feature values. The current state is investigated to check if there is a cluster including the current state. If there exists a cluster including the current state, the vertices of the cluster are provided to RQ-learning module in which they play a role of *neighboring states* in RQ-learning. However, if the current state is not included in any cluster, a *default action a* is performed to confirm that the reward of the current state can be estimated by a pre-specified *effect function* for the most similar cluster. Then, real reward r is received and utilized to obtain the similarity between the current state s_i and neighboring state $s_{i,j}$, where the reward of the neighboring state should be the one previously obtained by the *default action a*. The received reward is also fed back to clustering module. In clustering module, the current state and the received reward are exploited to generate clusters having similar reward distribution.

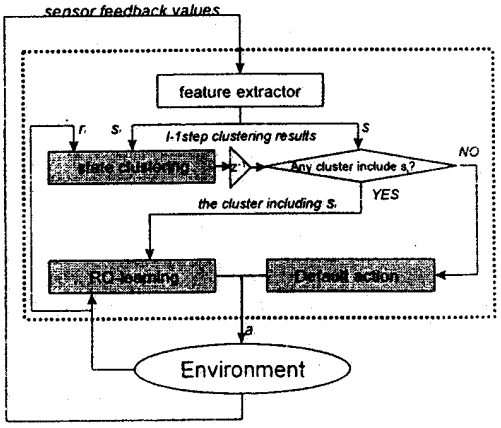


Fig. 6 Functional Architecture of CQ-learning system

It is remarked that there exist three nontrivial cases according to the existence of a cluster including the current state or the received reward. If there exists a cluster including the current state, and if the received reward is similar to that estimated by the *effect function* at the neighboring states, then region-based learning is applied to find the optimal action. And, if there exists a cluster including the current state, but if the received reward is much different from that estimated by such *effect functions*, the cluster is then contracted. Finally,

the current state is not included in any cluster, and the received reward is sufficiently similar to the reward estimated by all vertices of a cluster, clustering is carried out in such a way that the current state is made to be included in the most similar cluster.

4.2 State Clustering(SC) algorithm

There are two types of inputs for the state clustering module : state vector and scalar valued reward. If the input state belongs to a cluster, an action derived from the region-based Q-learning is applied. And a contraction process may be performed according to the reward for the action. To evaluate the current reward, a reward for current state is here estimated by using effect functions of neighboring states as follows;

$$\hat{r}_i = \frac{1}{N}\sum_{j=1}^{N}\mu_{i,j}(s_i, s_{i,j})\cdot r_j \,. \tag{16}$$

Next, the received real reward r_i is compared with the estimated reward $\hat{r}_i$ by using similarity function given as

$$\xi(r_i, \hat{r}_i) = \frac{1}{1+d^2(r_i, \hat{r}_i)}. \tag{17}$$

This similarity function plays a role of leading the cluster to have the same reward distribution property. If the similarity is lower than a threshold value, then the current cluster will be contracted in such a way that both the current state and the nearest neighboring state are separated from other neighboring states of the cluster as shown in Fig 7.

However, if the input state does not belong to any cluster, convex clustering process is executed to find the acceptable region of reward distribution. In convex clustering, reward r_{jd} for the default action of jth neighboring state is employed to compute M estimated rewards $\hat{r}_{id}^{j}$, $j=1,2,...,M$, for a current state as

$$\hat{r}_{id}^{j} = \mu_{i,j}(s_i, s_{i,j})\cdot r_{jd}, \quad j=1,2,...,M, \tag{18}$$

where M is the number of vertices of a cluster. Here, $\hat{r}_{id}^{j}$ is computed for all clusters. Then, the received real reward r_{jd} for a *default action* is compared with all estimated rewards. Similar clusters are determined if for every vertex in a cluster, similarity $\xi(r_{id}, \hat{r}_{id}^{j})$ is larger than a threshold value. This implies that the reward for a state inside a cluster can be estimated from vertices of the cluster (neighboring states). In case that there exist multiple clusters satisfying such a constraint, we will choose any one cluster as the most similar cluster.

It is remarked that for a input state, there may exist more than two clusters having similarity larger than the threshold value. In this case, the overlap of two clusters may be generated. Unlike the previous convex clustering[11], The proposed SC algorithm does not perform any special processing for the overlapping of clusters. A simple convex clustering example is shown in Fig. 7. For the details on the convex clustering, our previous work in [11] should be referenced.

4.3 Action generation

According to the results of clustering, there are two types of action generation. In case that a cluster including the current state exists, an proper action is generated by using the Region-based Q-learning method as described in sec 3. In other cases, a predefined action(*default action*) is performed and the state is clustered according to the received reward as described in 4.2.

5 Simulation Results

First, we simulated the comparison between the conventional Q-learning and CQ-learning in a 2-D feature space. The robot starts from an initial feature vector (50,50) and its goal is to reach target feature vector (350,350). The discrete resolution of each state axis for Q-learning is set at 35. 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 CQ-learning, the robot behaves well under such an optimal policy after 50 iterations. Fig. 8 shows the results of CQ-learning, where the region generated in CQ-learning guarantees that the reward to be assigned in this region can be predicted form the rewards of the *neighboring states*. As expected from Fig. 9, our CQ-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 it react to a current state with a better tuned action.

We have also simulated a visual tracking task of a two-link robot arm. For this, we utilized a 4-dimensional feature space which is consist of two joint angles of the 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}	velocity of link1
a_{l2}	velocity of link2

The reward used in this simulation is defined as

$$r = \frac{(\|x_{i+1} - x_g\| - \|x_i - x_g\|)}{K}, \quad (19)$$

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. At every sampling time, the robot is controlled by actions determined by our proposed CQ-learning technique to move toward the object as in Fig.10. In this simulation, performances for 3 different action ranges were investigated as shown in Fig. 10 (a), (b), and (c). From this simulation results, it is observed that the robot adaptively moves toward the object after 2300, 1750, and 1200 trials, respectively. Thus, it is believed that the expansion of action range does not make an abrupt increase of learning time when using CQ-learning.

6 Conclusion

In this paper, we proposed a novel method of Q-learning, where the method incorporated a region-based reward assignment being used to solve structural credit assignment problem and a convex clustering approach to find a region with the same reward attribution property. In order to show the validity of our method, we compared our method with conventional Q-learning method through a simple two dimensional free space navigation problem. The simulation results show that the required number of iterations needed to reach a goal using our method was much less than the conventional Q-learning. Also, we illustrated that our method was suitable for a visual tracking application that requires continuous states and actions in a real environment. For a real world application, we are currently working on the visual tracking experiment.

7 References

[1] M. A. Salichs, E. A. Puente, D. Gachet, and J. R. Pementel, "Learning behavioral control by reinforcement for an autonomous mobile robot," *Proc. IEEE Conference on R&A*, vol. 1, pp. 1436-1441, 1993.

[2] H. Berenji and 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," *Proc. the Ninth National Conference on Artificial Intelligence*, 1991.

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

[5] C. Watkins and 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] A. Minoru, U. Eiji, and H. Koh,"Behavior Coordination for a Mobile Robot Using Modular Reinforcement Learning," *Proc. IEEE/RSJ Conference on Intelligent Robots and Systems*, vol. 3, pp. 1329-1336, 1996.

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

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

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

[11] I. H. Suh, J. H. Kim, and F. J. H. Lee, "Fuzzy Clustering involving Convex Polytopes," *Proc. 5nd IEEE Conference on Fuzzy System*, New Orleans, LA, vol. 2, pp. 1013-1019,1996.

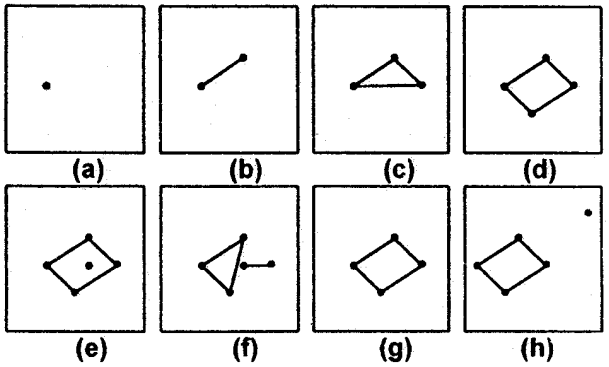
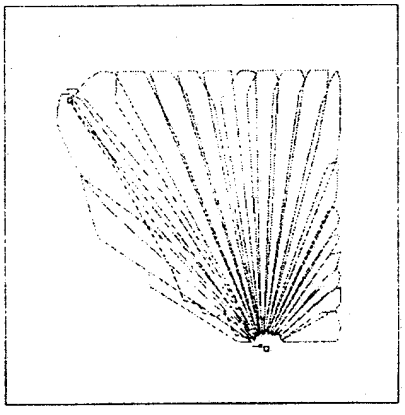
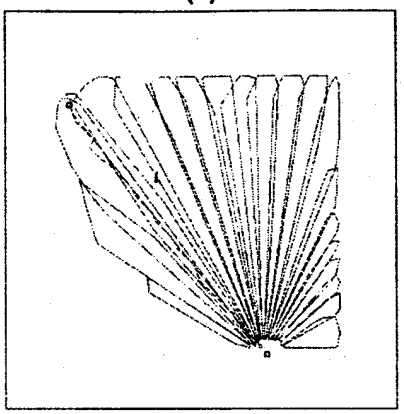


Fig. 7 Operation flow of SC Algorithm. (a) First, an input state arrives. (b),(c),(d) The most similar cluster expand to the current input state. (e) A new state is arrived in a the input state. (f) Since the similarity between the cluster and the input state is not acceptable, a new cluster is generated. (g) Since the similarity between the cluster and the input state is acceptable, any contraction does not bring about. (h) A new state is arrived, and since a new state is not similar to the nearest cluster, a new cluster is generated at the input state position.

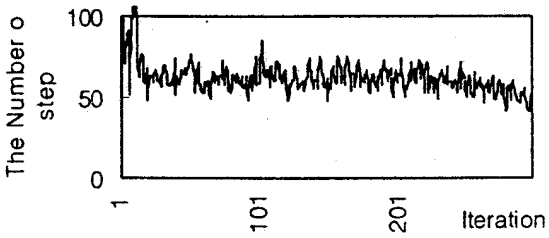




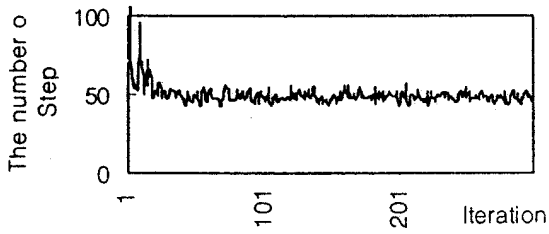
(a)

(b)

Fig. 8 CQ-learning simulation results: (a)Most clusters are found after 30th iteration (b)Each cluster converges to an optimal action after 50th iteration.

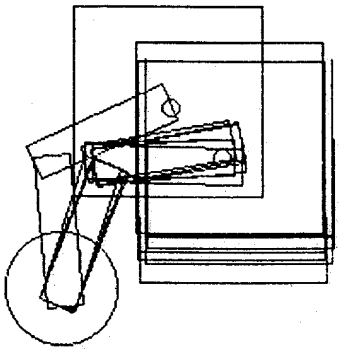


(a)

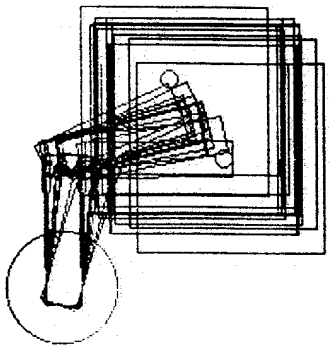


(b)

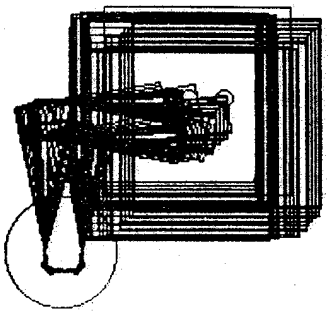
Fig. 9 The number of steps of Q, and CQ-learning: (a) Q-learning case, (b) CQ-learning case



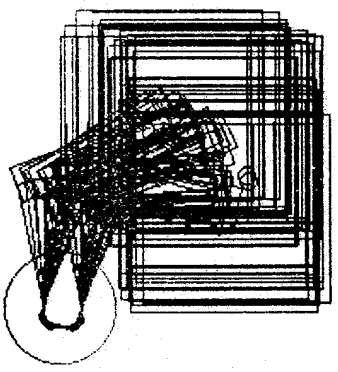
(a) Angular velocity Range(-15 ~ +15 degree/sampling time)



(b) Angular velocity Range(-10 ~ +10 degree/sampling time)



(c) Angular velocity Range (-5 ~ +5 degree/sampling time)



(d) Angular velocity Range (-10 ~ +10 degree/sampling time)

Fig. 10 Visual tracking of a 2-DOF SCARA robot by our proposed Cluster-based Q-learning technique: (a),(b),and, (c) Visual tracking an object at two different positions, (d) Visual tracking an object at arbitrary different positions