

# Behavior Coordination for a Mobile Robot Using Modular Reinforcement Learning

Eiji Uchibe, Minoru Asada, and Koh Hosoda

Dept. of Mech. Eng. for Computer-Controlled Machinery  
Osaka University, Suita, Osaka 565, Japan  
uchibe@robotics.ccm.eng.osaka-u.ac.jp

## Abstract

*Coordination of multiple behaviors independently obtained by a reinforcement learning method is one of the issues in order for the method to be scaled to larger and more complex robot learning tasks. Direct combination of all the state spaces for individual modules (subtasks) needs enormous learning time, and it causes hidden states. This paper presents a method of modular learning which coordinates multiple behaviors taking account of a trade-off between learning time and performance. First, in order to reduce the learning time the whole state space is classified into two categories based on the action values separately obtained by Q learning: the area where one of the learned behaviors is directly applicable (no more learning area), and the area where learning is necessary due to the competition of multiple behaviors (re-learning area). Second, hidden states are detected by model fitting to the learned action values based on the information criterion. Finally, the initial action values in the re-learning area are adjusted so that they can be consistent with the values in the no more learning area. The method is applied to one to one soccer playing robots. Computer simulation and real robot experiments are given to show the validity of the proposed method.*

## 1 Introduction

Realization of autonomous agents that organize their own internal structure in order to take actions towards achieving their goals is the ultimate goal of AI and Robotics. That is, the autonomous agents have to learn. Reinforcement learning has recently been receiving increased attention as a method for robot learning with little or no *a priori* knowledge and higher capability of reactive and adaptive behaviors [6]. In the reinforcement learning method, a robot and its environment are modeled by two synchronized finite state automata interacting in discrete time cyclical processes. The robot senses the current state of the environment and selects an action. Based on the state and the action, the environment makes a transition to a new state and generates a reward that is passed back to the robot. Through these interactions, the robot learns a purposive behavior to achieve a given goal.

Although the role of reinforcement learning is very

important to realize autonomous systems, the prominence of that role is largely dependent on the extent to which the learning can be scaled to solve larger and more complex robot learning tasks. Simple application of the reinforcement learning method to multiple robot tasks seems hard because of enormous amount of learning time. Modular learning is often used to cope with large scaled robot tasks.

Singh [11] defined a composite task as sequentially concatenating multiple elemental tasks, and rewards are generated only when the system achieves a subtask in a prescribed order. While, Whitehead et al. [14] proposed a modular architecture to coordinate multiple behaviors. Subtasks are independent of each other, and therefore their execution order can be arbitrary. The validity of these methods has been shown only by computer simulations where the action and state spaces are too idealized and the task seems simple and straightforward.

Connel and Mahadevan [5] proposed a rapid task learning for real robots by decomposing the whole task (box-pushing) into subtasks (finding, pushing a box, and unwedging) independent of each other. However, decomposition and switching conditions between subtasks are designed by the programmer. Gachet et al. [7] realized a coordinated behavior which is a linear combination of basic modules. However, the resultant behavior is not guaranteed as an optimal one.

Existing methods explained above assume that the subtask state spaces do not interfere with each other or they are completely independent of each other. This assumption is too idealized and often does not hold in real robot tasks.

Asada et al. [3] proposed a method for behavior coordination in a case that the subtask state spaces interfere with each other, and they applied it to real soccer robots. They reused the learned action values as initial knowledge in re-learning process to reduce the time, but still they suffer from enormous amount of the learning time. Since hidden states are given by the programmer, the robot cannot cope with unexpected hidden states. Maybe, they need some methods of hidden state detection such as [4, 9].

In this paper, we propose a method of modular learning which coordinates multiple behaviors taking account of a trade-off between learning time and per-

formance. First, in order to reduce the learning time the whole state space is classified into two categories based on the action values separately obtained by Q learning: the area where one of the learned behaviors is directly applicable (no more learning area), and the area where learning is necessary due to the competition of multiple behaviors (re-learning area). Second, hidden states are detected by model fitting to the learned action values based on the information criterion. Finally, the initial action values in the re-learning area are adjusted so that they can be consistent with the values in the no more learning area. The method is applied to one to one soccer playing robots. Computer simulations and real robot experiments are given to show the validity of the proposed method.

## 2 Reinforcement Learning

### 2.1 Q learning

Q learning is a form of model-free reinforcement learning based on stochastic dynamic programming. It provides robots with the capability of learning to act optimally in a Markovian environment [12]. We assume that the robot can discriminate the set  $\mathbf{S}$  of distinct world states, and can take the set  $\mathbf{A}$  of actions on the world. A simple version of a Q learning algorithm used here is shown in Fig.1.

1. Initialize  $Q(s, a)$  to 0 for all state  $s$  and action  $a$ .
2. Perceives current state  $s$ .
3. Choose an action  $a$  according to action value function.
4. Carry out action  $a$  in the environment. Let the next state be  $s'$  and immediate reward be  $r$ .
5. Update action value function from  $s, a, s'$ , and  $r$ ,

$$Q_{t+1}(s, a) = (1 - \alpha_t)Q_t(s, a) + \alpha_t(r + \gamma \max_{a' \in \mathbf{A}} Q_t(s', a')) \quad (1)$$

where  $\alpha_t$  is a learning rate parameter and  $\gamma$  is a fixed discounting factor between 0 and 1.

6. Return to 2.

**Fig.1** A simple version of the 1-step Q learning algorithm.

### 2.2 Applying Q Learning to Multiple Goal Tasks

The time needed to acquire an optimal policy mainly depends on the size of state space. If we apply the monolithic Q learning into multiple goal tasks, the expected learning time is exponential in the size of state space [13]. Therefore, a number of methods have been

employed to speed up learning in multiple tasks. One technique is to divide a multiple task into some sub-tasks and coordinate behaviors which is independently acquired. We briefly survey the method by Asada et al. [3].

#### (a) Simple summation of action value functions

The action value function  $Q_{ss}(s, a)$  for the coordinated behavior is given by

$$Q_{ss}(s, a) = \sum_{i=1}^n {}^i Q({}^i s, a). \quad (2)$$

In this scheme, the selected action sometimes might not make any sense because the simple sum cannot consider the combined situations, and also easily trapped into the local maxima.

#### (b) Switching action value functions

The action value function  $Q_{sw}(s, a)$  for the coordinated behavior is given by

$$Q_{sw}(s, a) = {}^i Q({}^i s, a) \quad \text{in some situations,} \quad (3)$$

It seems hard to appropriately determine the situations to switch the functions. Therefore, we need a carefully designed decision rule to switch the policies. The following method provides this rule by learning a new policy coping with new situations.

#### (c) Using learning results as initial value of a new action value function

In the above methods, the previously learned action value functions are simply summed or switched. Therefore, these method cannot cope with local maxima and/or hidden states caused by combination of s-tate spaces. Consequently, an action suitable for these situations has never been learned. To cope with these new situations, the robot needs to learn a new behavior by using the previously learned behaviors. The method is as follows;

1. Construct a new state space  $\mathbf{S}$ :
  - (a) construct the directly combined state space.
  - (b) find such states that are inconsistent.
  - (c) resolve the inconsistent states by adding new sub-states  $s_{sub} \in \mathbf{S}$ .
2. Learn a new behavior in the new state space  $\mathbf{S}$ :
  - (a) use the values of the action value function  $Q_{ss}$  as the initial values of  $Q_{rl}$  for both the normal states  $s$  and the new sub-states  $s_{sub}$ . For the new sub-states, we use the original value of  $Q_{ss}(s, a)$  before generating these new states. That is,

$$\begin{aligned} Q_{rl}(s, a) &= Q_{ss}(s, a) \\ Q_{rl}(s_{sub}, a) &= \text{original value of } Q_{ss}(s, a) \end{aligned} \quad (4)$$

- (b) control the strategy for the action selection in such a way that conservative strategy is used around the normal states  $s$  and high random strategy around the new sub-states  $s_{sub}$  in order to reduce the learning time.

### 2.3 Problems of the method in [3]

In our previous work [3], the two problems are left unsolved. One is that the robot has to learn at the all states. Though conservative strategy is used around the normal states, the robot sometimes executes an inappropriate action because of exploration. This causes explosion of learning time. The other is that inconsistent states are found by the programmer. If the programmer misses inconsistent states, the robot can not acquire a suitable behavior.

In order to overcome these two problems, we present a method for each. For the former, modular learning which coordinates multiple behaviors taking account of a trade-off between learning time and performance is proposed in section 3. For the latter, model fitting to the learned action values based on information criterion is proposed so as to detect hidden states in section 4.

## 3 Modular Reinforcement Learning

In this section, we present a method of “modular reinforcement learning” to coordinate multiple behaviors when action value functions of subtasks are given to the robot.

### 3.1 Module Construction

**Fig.2** shows an overview of module construction procedure. First, a new state space  $\mathcal{S}$  is composed by direct product of state spaces corresponding to  $n$  modules, where  $n$  is two for the sake of reader’s understanding. We define the kernel state of each module in order to prepare for clustering the composite state space.

Let us define the following:

- ${}^i T$  : a subtask corresponding the  $i$ -th module, ( $i = 1, \dots, n$ )
- ${}^i \mathcal{S}$  : state space for the  $i$ -th subtask,
- ${}^i s_k$  : the  $k$ -th state of the  $i$ -th state space  ${}^i \mathcal{S}$ ,
- ${}^i Q$  : action value function of the  $i$ -th subtask,
- $\mathcal{S}$  : a composite state space obtained by direct combinations of  $n$  state spaces,
- $s$  : one sub-state in the composite state space  $\mathcal{S}$ .

We define the maximum action value function  ${}^i q_{max}^k$  in terms of action.  ${}^i q_{max}^k$  is computed by

$${}^i q_{max}^k({}^i s_k) = \max_{b \in \mathbf{A}} {}^i Q({}^i s_k, b),$$

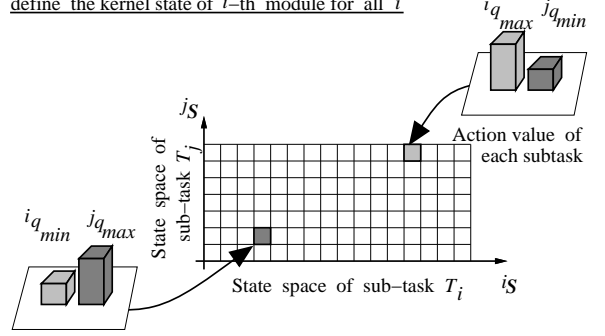
for a state  ${}^i s_k \in {}^i \mathcal{S}$  of subtask  $T_i$ . If

$$s^i = \arg \max_{{}^i s_k \in {}^i \mathcal{S}} {}^i q_{max}({}^i s_k), \text{ and} \quad (5)$$

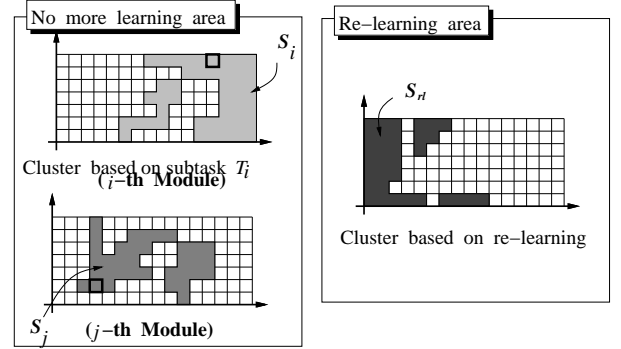
$$s^i = \arg \min_{j s_k \in j \mathcal{S}} {}^j q_{max}({}^j s_k) \text{ for all } j \neq i, \quad (6)$$

we regard the state  $s^i$  as the kernel state  $s_{kernel}^i$  of the  $i$ -th module.

1. Construct the directly combined state space  ${}^c \mathcal{S}$  and, define the kernel state of  $i$ -th module for all  $i$



2. Classify  $\mathcal{S}$  into clusters by ISODATA



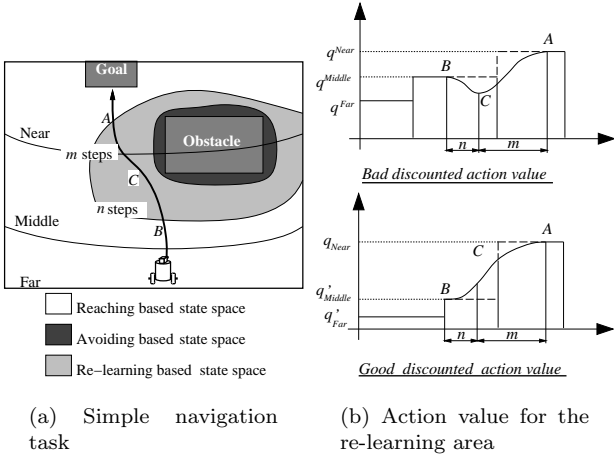
**Fig.2** An overview of module construction procedure

Then, in order to reduce the learning time, the whole state space is classified into two categories based on the maximum action values separately obtained by Q learning: the area where one of the learned behaviors is directly applicable (no more learning area), and the area where learning is necessary due to the competition of multiple behaviors (re-learning area). Then, all states  $s \in \mathcal{S}$  are classified according to the Mahalanobis distance between the non-kernel state  $s$  and the kernel states  $s_{kernel}$ . We apply ISODATA clustering algorithm to classify these action values. The ISODATA is an iterative and non-hierarchical clustering method. Eventually composite state space  $\mathcal{S}$  is classified into the no more learning area  $\mathcal{S}_i$ ,  $i = 1 \dots n$  and the re-learning area  $\mathcal{S}_{rl}$ . These area are exclusive.

### 3.2 Learning Schema

If both the current state  $s$  and the transited state  $s'$  belong to the re-learning area, we can apply normal Q learning. On the other hand both  $s$  and  $s'$  belong to the no more learning area, we do not need to update action value functions any more.

The problem is the estimation of discounted sum of the reward to update action value function, if  $s$  belong to the re-learning area while  $s'$  belong to the no more learning area. In general, action values before and after coordination might not be consistent



**Fig.3** Unbalance problem between action value function of modules

between different areas. Because, action values before coordination are acquired independently by different subtasks, and therefore direct use of action values simply brings to local maxima. Suppose that the simple navigation task as shown in **Fig.3(a)** is given to the robot, this task is decomposed into two subtasks (reaching and avoidance), and the optimal behaviors for both subtasks are obtained by Q learning independently.

Suppose that we simply applied Q learning with inadequate discount factor  $\gamma_c$  for coordination, and let  $q^{\text{near}}$  and  $q^{\text{middle}}$  be action value of Near and Middle respectively.

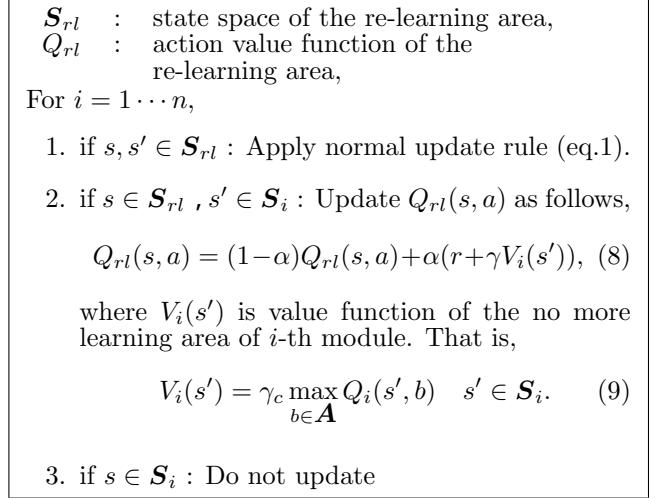
$$\begin{aligned} q(C \rightarrow A) &= \gamma_c^m q^{\text{near}}, \\ q(C \rightarrow B) &= \gamma_c^n q^{\text{middle}}, \end{aligned}$$

where  $q(C \rightarrow A)$  and  $q(C \rightarrow B)$  denote action value from C to A and from C to B respectively. If  $q(C \rightarrow A) < q(C \rightarrow B)$ , the robot might move back from C to B because of an inadequate discount factor  $\gamma_c$  (See **Fig.3(b)**).

Therefore, we have to adjust the action value function. For an optimal policy of  $i$ -th subtask  $i f_{opt}^*$ , calculate  $steps(i, s)$ , the physical number of steps to goal, given that the process begins in state  $i, s$  and follows optimal policy thereafter. Estimated action value functions are appropriately discounted using  $steps(i, s)$  as a discount factor. For example,  $\gamma_c$  of **Fig.3** is calculated as follows:

$$\gamma_c = E \left\{ \sqrt{q^{\text{near}} - q^{\text{middle}}} \right\}^{m+n}. \quad (7)$$

If  $\gamma_c \geq 1$ ,  $\gamma_c$  is adjusted to about 1.0. Eventually, the action value function  $Q_{rl}$  can be defined as shown in **Fig.4**.



**Fig.4** Update rule of module learning

## 4 Hidden States Detection based on A-IC

In the above method, hidden(inconsistent) states are not considered when some behaviors (result of learning) are combined. The hidden states prevent the learning robot from acquiring an optimal behavior, therefore the robot should be able to find hidden states autonomously. In this section, we present a statistical method to detect hidden states recursively.

Let  $\{q_t\}$  be a time series data of action value  $Q(s, a)$  in state  $s$  and action  $a$ , and suppose that fitting the transition of action value into  $(p, q)$  dimensional autoregressive moving average model (ARMA( $p, q$ )) which is computed by

$$q_t + \sum_{i=1}^p a_i q_{t-i} = e_t + \sum_{i=1}^q b_i e_{t-i}, \quad (10)$$

where  $a_i$  and  $b_i$  are the parameters of AR and MA respectively, and  $e_t$  is the noise and

$$E\{e_t\} = 0, \quad E\{e_t e_{t-\tau}\} = \sigma_e^2 \delta_{t\tau},$$

where  $\sigma_e^2$  is variance of  $\{e_t\}$ . The problem is how to determine the dimension  $p$  and  $q$  of ARMA model. Here, we apply AIC (Akaike's Information Criterion), which is widely used in the field of time series analysis [1] to determine  $p$  and  $q$ . For some  $p$  and  $q$ ,  $a_i$  and  $b_i$  are calculated by *prediction error method*(PEM) and then AIC is computed by

$$AIC = N \ln \hat{\sigma}_e^2 + 2(p + q), \quad (11)$$

where  $N$  is the number of data sets, and the factors unrelated to comparison are ignored. The estimation

of  $\sigma_e^2$ , and  $\hat{\sigma}_e^2$  is calculated by

$$\hat{\sigma}_e^2 = \frac{1}{N} \sum_{t=1}^N \hat{e}_t^2, \quad (12)$$

where  $\hat{e}_t^2$  is the estimation of prediction error. The dimension  $p$  and  $q$  of ARMA model is determined by minimizing AIC.

If the state  $s$  is not a hidden state and the Q learning algorithm converges, the dimension of model should be 0. However if  $s$  is a hidden state the dimension is not equal to 0. Hence if the dimension of ARMA model is more than 2, then the state  $s$  is regarded as a hidden state. Hereafter the action value  $Q(s, a)$  for hidden state  $s$  are deleted, tuple  $\{s_{prev}, a_{prev}, s\}$  is added to the composite state space as one state, where  $s_{prev}$  and  $a_{prev}$  denote the old state and the action which is executed in state  $s_{prev}$  and makes transition to  $s$ . Then initialize action value to old  $Q(s, a)$ . The simple outline is given in Fig.5.

For the time series data of action value,

1. Fit the transition into ARMA( $p, q$ ).
2. Compute AIC.
3. Judge if the state  $s$  would be hidden states based on AIC.

**Fig.5** Detection procedure of hidden states by autoregressive model

## 5 The Task and Assumptions

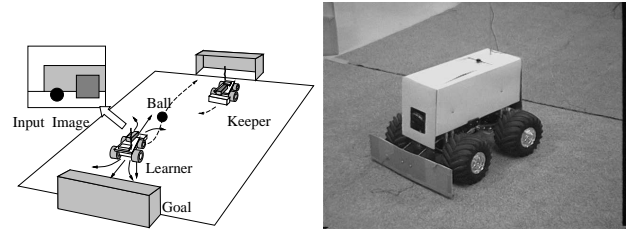
### 5.1 The Multiple Task

The task for a mobile robot is the same as [3], which is to shoot a ball into a goal without collisions with a keeper robot (see Fig.6(a)). The environment consists of a ball, two goals, four lines and a keeper robot, and the each mobile robot has a single color TV camera. The robot does not know the location and the size of the objects, the size and the weight of the ball, keeper robot, any camera parameters such as focal length and tilt angle, nor kinematics/dynamics of itself. Our mobile robot moves around using a 4-wheel steering system.

The effects of an action against the environment can be informed to the robot only through the visual information. Here, we consider to divide the task into two subtasks; one is to shoot a ball into the goal which has been learned in [2] and the other is to avoid a moving keeper robot. Then, we coordinate these learned behaviors into one.

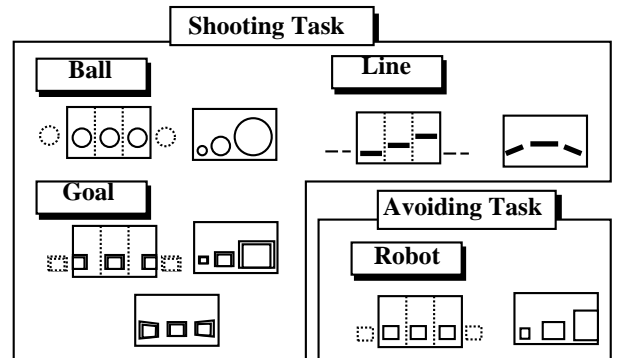
### 5.2 Construction of State and Action Spaces

In a shooting subtask, the state space consists of ball image, goal image and line image, while in an avoiding



(a) A robot environment

(b) Our soccer robot



(c) The components of state spaces

**Fig.6** The task is to shoot a ball into the goal avoiding collisions with a keeper robot.

subtask, the state space consists of only keeper robot image (See Fig.6(c)).

As motor commands, we have 7 actions such as go straight, turn right, turn left, stop, go backward, as shown in Fig.6(a). In real robot tasks, one physical action does not always cause a state transition, therefore the learning often does not work well. This is called “state-action deviation” problem [2]. To avoid this problem, we define action as a sequence of action primitives from 7 physical actions until state transition happens.

### 5.3 Reward Function and Discounting Factor

In shooting subtask, we assign a reward value 1 when the ball was kicked into the goal or 0 otherwise. On the other hand, a reward value  $-0.3$  is given to the robot when a collision between two robots is happened in avoiding subtask. In modular learning, same reward function is used.

Discounting factors are 0.9 and  $-0.1$  in shooting and avoiding subtask respectively.  $\gamma$  for coordination is estimated as described in 3.2.

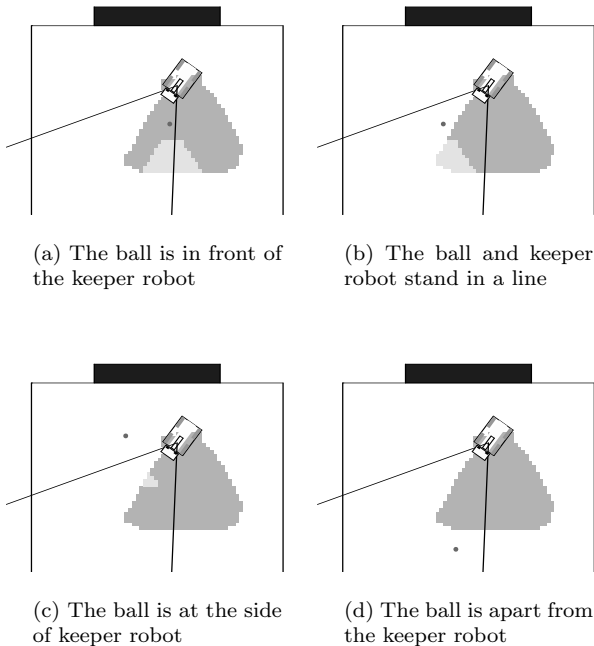


Fig.7 Clustering results

## 6 Experiments

The experiment consists of two parts: first, learning the optimal policy through computer simulations, then apply the learned policy to a real situation.

### 6.1 Computer Simulations

First, the examples of the classified states are shown in Fig.7, where a keeper robot, a ball, two goals, and four lines are displayed. The dark gray region corresponds to the re-learning area, and the light gray one corresponds to the no more learning area for avoiding behavior. The rest is the no more learning area for shooting behavior. For example, the re-learning area in Fig.7(a) is larger than one in Fig.7(b) because the ball is close to the goal keeper. Figs.7(c) and (d) show cases when one behavior can be directly applicable because of almost no interference between multiple behaviors.

Next, we show the average performance over 10 runs for many coordination method in Table 1, where avoidance (shooting) switching has a priority over shooting (avoidance) behaviors. As a matter of course, re-learning method is superior to all other methods of learning in Table 1. In particular, we focus on the differences in performance between learning all states described in [3] and modular reinforcement learning in Table 2. However the learning time needed to converge of that method is about 5.6 times as long as the modular learning. Because the upper limit of success rate of shooting is bound, we can not conclude that modular reinforcement learning is the best

Table 1 Average performance over 10 runs measured

Integration method	success of shooting(%)	mean steps to collision	mean steps to shooting
only shooting	50.3	62.5	131.2
simple summation	36.1	172.3	231.2
avoidance switching	39.5	6207.4	414.4
shooting switching	49.6	95.2	273.5
relearning	60.8	5048.5	128.3
modular learning	57.3	3624.8	138.6

Table 2 Average performance over 10 runs measured *relearning vs. modular learning*

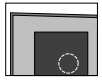

performance	all states	module
success rate of shooting (%)	60.8	57.3
steps to collision	5048.5	3624.8
steps to shoot	128.3	138.6
convergence time	718.5	128.9
# of states of learning	11132	395

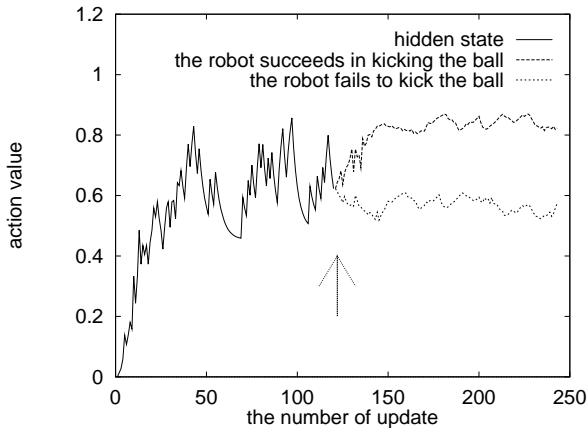
method completely, but modular reinforcement learning are effective in regard to the trade-off between performance and learning time.

Table 3 shows the two typical examples of statistical detection of hidden states. One is the case the ball is occluded by the keeper robot, which is a result of direct combination of state spaces. The other is the case that the state space is too coarse for the states and actions to correspond to one to one. If the robot succeeds in kicking the ball, then the action value becomes high because it is easy to shoot the ball into the goal. However if the robot fails to kick the ball, the environment makes a transition to a hidden state such as “ball is lost into left”, therefore the action value becomes low. Consequently the variance of action value is large. Fig.8 shows that the transition of the state at the right side in Table 3 becomes stable after that this state is found “hidden” and a new state is added (an arrow indicates the time of hidden state detection).

Fig.9 shows a sequence of shooting behavior by the modular learning method. In these figures, the learning robot and the keeper robot are colored in black and white, respectively. The dotted lines emerged from them shows their visual angles. The opponent tries to chase after the robot with the probability of 60% as long as it can see the robot. Otherwise, it stops.

**Table 3** Typical hidden states detected by AIC-based fitting

state		
dimension of ARMA model( $p$ )	1	2



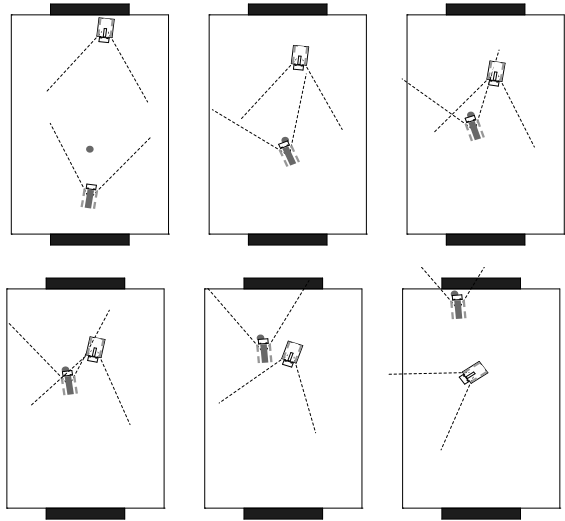
**Fig.8** Transition of action value of hidden states detected by ARMA model enhanced by *AIC*

## 6.2 Real System

**Fig.10** and **Fig.11** show a configuration of the real mobile robot, and the example of input images taken from a learning robot (left) and a goal keeper (right). The image taken by a TV camera mounted on the robot is transmitted to a UHF receiver and processed by Datacube MaxVideo 200, a real-time pipeline video image processor. In order to simplify and speed up the image processing time, we painted the ball, the goal, and the enemy in red, blue, and yellow, respectively. The input NTSC color video signal is first converted into HSV color components in order to make the extraction of the objects easy. We have constructed the radio control system of the robot, following the remote-brain project by Inaba et al. [8]. The image processing and the vehicle control system are operated by VxWorks OS on MC68040 CPU which are connected with host Sun workstations via Ethernet. The tilt angle is about  $-26$  [deg] so that robot can see the environment effectively. The horizontal and vertical visual angle are about 67 [deg] and 60 [deg], respectively. **Fig.12** shows a sequence of images where the robot achieved the goal avoiding a keeper robot.

## 7 Discussion and Future Works

This paper presents a method of modular reinforcement learning which coordinates multiple behaviors



**Fig.9** The robot succeeded in shooting a ball into a goal avoiding a moving keeper robot.

taking account of a trade-off between learning time and performance and a method of autonomous detection of hidden states which fits model to the time series action values based on the information criterion, and demonstrates experiments on a simulated and a real robot.

In our method, we need an estimation procedure of average steps to the goal state in order to adjust action value functions. It does require that average steps to goal be computed after convergence because of “state-action deviation” problem. Instead, R learning [10] which maximize average reward but discounted cumulative reward might be promising.

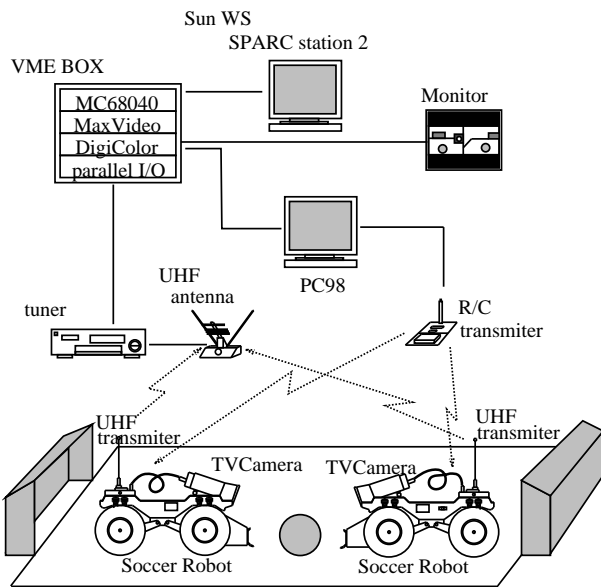
As future work we hope to challenge topics as follows: 1. *autonomous decomposition of multiple tasks*, 2. *simultaneous learning of multiple robots*. Our final goal is to build up a team of soccer playing robots in which not only the learning from competitive agents but from cooperative agents as well should be studied to realize team plays such as centering, passing, and so on.

## Acknowledgment

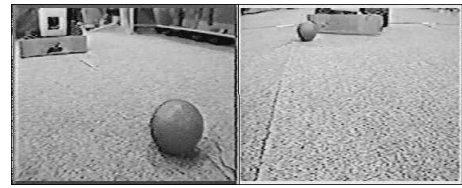
The authors thank Mr. Masateru Nakamura for their efforts in implementing a real robot system.

## References

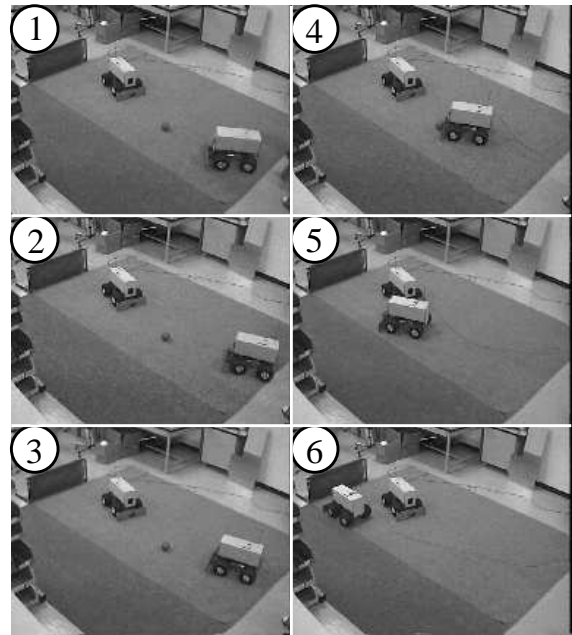
- [1] H. Akaike. A New Look on the Statistical Model Identification. *IEEE Trans. AC-19*, pp. 716–723, 1974.
- [2] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda. Vision-Based Reinforcement Learning for Purposeful Behavior Acquisition. In *Proc. of IEEE Inter-*



**Fig.10** A configuration of the real system.



**Fig.11** Real input images



**Fig.12** The robot succeeded in shooting a ball into a goal avoiding a stationary keeper robot.

*national Conference on Robotics and Automation*, pp. 146–153, 1995.

[3] M. Asada, E. Uchibe, S. Noda, S. Tawaratsumida, and K. Hosoda. Coordination Of Multiple Behaviors Acquired By A Vision-Based Reinforcement Learning. In *Proc. of the 1994 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 2, pp. 917–924, 1994.

[4] L. Chrisman. Planning for Closed-Loop Execution Using Partially Observable Markovian Decision Processes. In *AAAI Spring Symposium Series: Control of Selective Perception*, 1992.

[5] J. H. Connel and S. Mahadevan. Rapid Task Learning for Real Robot. In *Robot Learning* [6], chapter 5, pp. 105–140.

[6] J. H. Connel and S. Mahadevan. *Robot Learning*. Kluwer Academic Publishers, 1993.

[7] D. Gachet, M. A. Salichs, L. Moreno, and J. R. Pimentel. Learning Emergent Tasks for an Autonomous Mobile Robot. In *Proc. of the 1994 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 290–297, 1994.

[8] M. Inaba. Remote-Brained Robotics : Interfacing AI with Real World Behaviors. In *Preprints of ISRR'93*, Pitsuburg, 1993.

[9] R. A. McCallum. Instance-Based Utile Distinctions for Reinforcement Learning with Hidden State. In *Proc. of the 12th International Conference on Machine Learning*, pp. 387–395, 1995.

[10] A. Schwartz. A reinforcement learning method for maximizing undiscounted rewards. In *Proc. of the 10th International Conference on Machine Learning*, pp. 298–305, 1993.

[11] S. P. Singh. Transfer of Learning by Composing Solution of Elemental Sequential Tasks. In *Machine Learning*, Vol. 8, pp. 99–115, 1992.

[12] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, University of Cambridge, May 1989.

[13] S. D. Whitehead. Complexity and Coordination in *Q-Learning*. In *Proc. of the 8th International Workshop on Machine Learning*, pp. 363–367, Evanston, IL, 1991. Morgan Kaufmann.

[14] S. D. Whitehead, J. Karlsson, and J. Tenenber. Learning Multiple Goal Behavior Via Task Decomposition And Dynamic Policy Merging. In Connel and Mahadevan [6], chapter 3.