Vision-Based Reinforcement Learning for RoboCup : Towards Real Robot Competition

Eiji Uchibe, Minoru Asada, Shoichi Noda,
Yasutake Takahashi and Koh Hosoda

Dept. of Mechanical Engineering for Computer-Controlled Machinerey
Osaka University, Suita, Osaka 565 Japan
uchibe@robotics.ccm.eng.osaka-u.ac.jp

Abstract

We have been doing a research on vision-based reinforcement learning and applied the method to build real soccer playing robots towards **RoboCup Initiative**. In the first stage [2, 4], a robot learned to shoot a ball into a goal given the state space in terms of the sizes and the positions of both the ball and the goal in image. In the second stage [4], we set up an opponent just before the goal, that is, a goal keeper, and make the robot learn to shoot a ball into a goal avoiding the goal keeper. The behavior of the opponent is scheduled for the learner to efficiently obtain the desired behavior [3]. This paper describes several research issues for **RoboCup** with real robots along with our research projects.

1 Introduction

Building robots that learn to perform a task has been acknowledged as one of the major challenges facing AI and Robotics. 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 scheme, a robot and an environment are modeled by two synchronized finite state automatons 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. Many researchers in the field of machine learning have been concerned with the convergence time of the learning, and have developed methods to speed it up. They have also extended these techniques from solving single goal tasks to multiple goal ones. However, almost all of them have only shown computer simulations in which they assume ideal sensors and actuators, where they can easily construct the state and action spaces consistent with each other. A typical example is the 2-D grid environment in which the robot can take an action of going forward, backward, left, or right, and its state is encoded by the coordinate of the grid (i.e., an absolute (global) positioning system is assumed). Although the uncertainties of sensor and actuator outputs are considered by a stochastic transition model in the state space, such a model cannot account for the accumulation of sensor errors in estimating the robot position. Further, from the viewpoint of real robot applications, we should construct the state space so that it can reflect the outputs of the physical sensors which are currently available and can be mounted on the robot.

Mahadevan and Connel [5] proposed a method of rapid task learning on a real robot. They separated a pushing task into three subtasks of "finding a box", "pushing a box", and "getting unwedged", and applied Q learning, a widely used reinforcement learning method, to each of them. Since only proximity sensors such as bumper and sonar sensors are used, the acquired behaviors are limited to local ones and therefore these behaviors are not suitable for more global and goal-directed tasks such as carrying a box to a specified location. For such tasks, visual sensors could be more useful because they might be able to capture the image of the goal in a distant place. However, there are very few examples of use of visual information in reinforcement learning, probably because of the cost of visual processing.

As a test bed for real robot applications of the reinforcement learning method, we have selected soccer playing robots because building such a system includes various kinds of aspects of fundamental AI problems. One can see the details of this reason in [8]. In this paper, we show our research projects along with research issues involved in realizing **RoboCup Initiative** with real robots. They are

- mechanical design and system suitable for various kinds of plays,
- real time sensing capability mounted on each player, and
- learning capability coping with various formation

of team playing.

The remainder of this article is structured as follows: In the next section, we give a brief overview of Q learning and explain about a real robot system to test our method. Next, we show a soccer robot that learns how to shoot a ball into a goal using the Q learning method based on only visual information. In the first stage [2], we prepared the state space in terms of the sizes, positions, and the orientation of the ball and the goal in image captured by the robot, and the action space in which the robot can send one of motor commands (forward, stop, and backward motions) into two independent motors. In the second stage [4], we set up an opponent just before the goal, that is, a goal keeper, and make the robot learn to shoot a ball into a goal avoiding the goal keeper. The behavior of the opponent is scheduled for the learner to efficiently obtain the desired behavior [3]. We do not mention recent topics [1, 9, 10] because of no space.

2 Q learning

Before getting into the details of our system, we will briefly review the basics of Q learning. For a more thorough treatment, see [11]. The general reinforcement learning problem is typically stated as finding a policy that maximizes discounted sum of the reward received over time. A policy f is mapping from $\boldsymbol{S}$ to $\boldsymbol{A}$. This sum called the *return* and is defined as:

$$\sum_{n=0}^{\infty} \gamma^n r_{t+n}, \tag{1}$$

where r_t is the reward received at step t given that the agent started in state s and executed policy f. γ is the discounting factor, it controls to what degree rewards in the distant future affect the total value of a policy and is just slightly less than 1.

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 [11]. We assume that the robot can discriminate the set $\boldsymbol{S}$ of distinct world states, and can take the set $\boldsymbol{A}$ of actions on the world. A simple version of a Q learning algorithm used here is shown in Fig.1.

3 Real Robot System

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, or 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.

Fig.2 shows a configuration of the real mobile robot system.

We have constructed the radio control system of the robot, following the remote-brain project by Inaba et

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,

$$\begin{aligned} Q_{t+1}(s,a) &= (1-\alpha_t)Q_t(s,a) \\ &\quad +\alpha_t(r+\gamma \max_{a' \in \boldsymbol{A}} Q_t(s',a')) \end{aligned} \tag{2}$$

   where r is the actual reward value received for taking action a in a situation s, α_t is a learning rate parameter and γ 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.

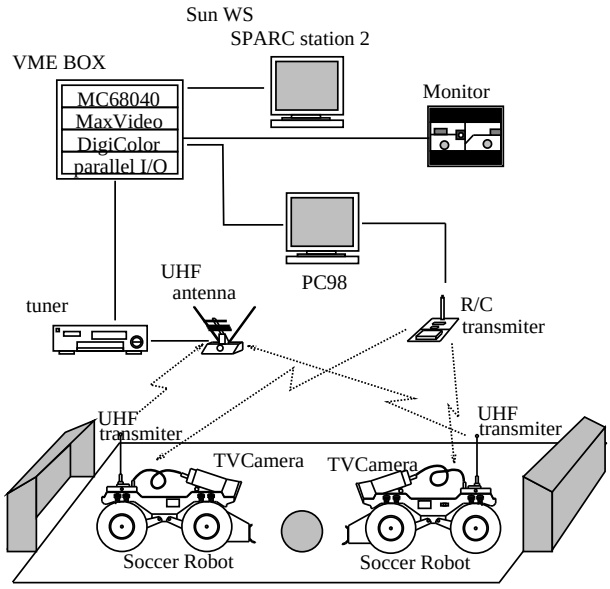

Fig.2 A configuration of the real system

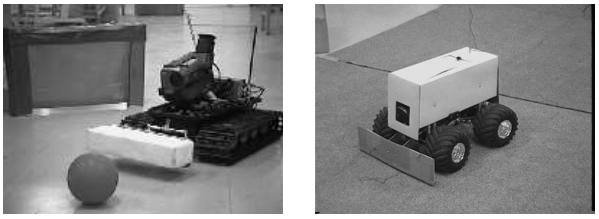

Fig.3 A picture of the radio-controlled vehicle

al. [7]. Two pictures of the real robot are shown in **Fig.3**.

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. The image processing and the vehicle control system are operated by VxWorks OS on MC68040 CPUs which are connected with host Sun workstations via Ether net. In order to simplify and speed up the image processing time, we painted the ball, the goal, and the opponent 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.

4 Shooting Behavior Acquisition [2]

4.1 Task, Assumptions

In this section, the task for a mobile robot is to shoot a ball into a goal as shown in **Fig.4**. The problem we address here is how to develop a method which automatically acquires strategies for doing this. In shooting task, we assume that the environment consists of a ball and a goal.

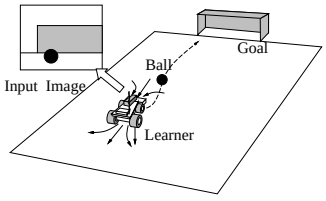

Fig.4 The task is to shoot a ball into a goal

4.2 Construction of State and Action Spaces

Fig.5 shows sub-states of ball and goal in which the position and the size of the ball or goal are naturally and coarsely classified into each state. Due to the peculiarity of visual information, that is, a small change near the observer results in a large change in the image and a large change far from the observer may result in a small change in the image, one action does not always correspond to one state transition. We call this the **"state-action deviation problem"**: **Fig.6** indicates this problem, the area representing the state

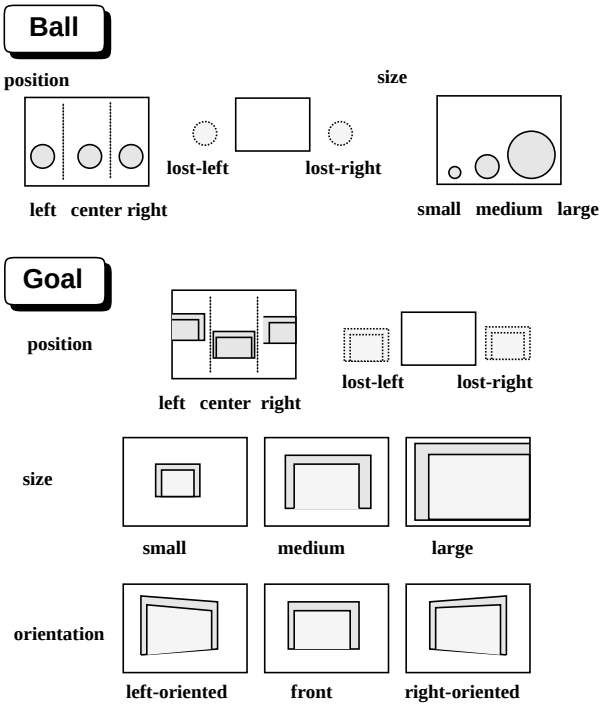

Fig.5 The ball sub-states and the goal sub-states

"the goal is far" is large, therefore the robot frequently returns to this state if the action is forward. This is highly undesirable because the variations in the state transitions is very large, consequently the learning does not converge correctly.

To avoid this problem, we reconstruct the action space as follows. Each action is regarded as an action primitive. The robot continues to take one action primitive at a time until the current state changes. This sequence of the action primitives is called an action. In the above case, the robot takes a forward motion many times until the state "the goal is far" changes into the state "the goal is medium".

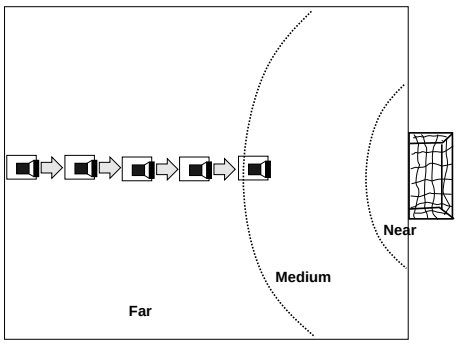

Fig.6 A state-action deviation problem

4.3 Learning from Easy Missions

In order to improve the learning rate, the whole task was separated into different subtasks in [5]. By contrast, we do not decompose the whole task into subtasks of finding, dribbling, and shooting a ball. Instead, we first used a monolithic approach. That is, we place the ball and the robot at arbitrary positions. In almost all the cases, the robot crossed over the field line without shooting the ball into the goal. This means that the learning did not converge after many trials. This is the famous *delayed reinforcement* problem due to no explicit teacher signal that indicates the correct output at each time step. To avoid this difficulty, we construct the learning schedule such that the robot can learn in easy situations at the early stages and later on learn in more difficult situations. We call this *Learning from Easy Missions* (or LEM).

5 Shooting a Ball with Avoiding an Opponent [3, 4]

In the second stage, we set up an opponent just before the goal, that is, a goal keeper, and make the robot learn to shoot a ball into a goal avoiding the goal keeper (See Fig.7). The basic idea is first to obtain the desired behavior for each subtask, and then to coordinate two learned behaviors. For the first subtask (shooting behavior), we have already obtained the learned policy by using the state space shown in 5. For the second subtask (avoiding behavior), we add the sub-states for the opponent that consist of the size and position of it in image.

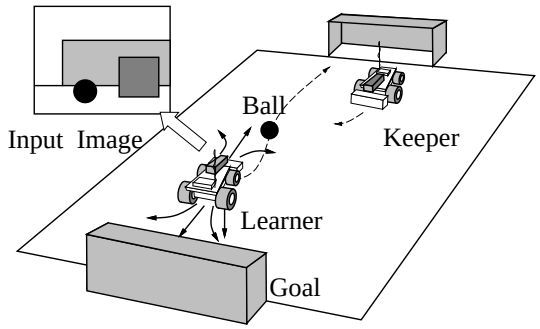

Fig.7 The task is to shoot a ball into the goal avoiding an opponent.

We assign a reward value 1 when the ball is entered into the goal or 0 otherwise for the shooting task, and -0.3 when a collision with a moving obstacle occurs. A discounting factor γ^g is used to control to what degree rewards in the distant future affect the total value of a policy. In the shooting task, we set the value a slightly less than 1 ($\gamma^g = 0.9$), and for the avoiding task $\gamma^r = 0.1$.

5.1 Learning a Reflexive Behavior

The Q learning method can obtain not only goal-directed behaviors but also reflexive ones as well by slightly changing some parameters and updating equations. Unlike the goal-directed behaviors to find the path from the current state to the goal state, reflexive behaviors are reactive, and therefore, the discounting factor γ^r should be much smaller so that the action-value for the distant future action cannot be affected.

A typical example of such behaviors is "collision avoidance" which has another different property from that of goal-directed behaviors. That is, any action can be allowed to be taken unless it causes collisions with other objects (agents). In order to learn such a behavior, the negative reward should be assigned for the state-action pair which causes a collision with a moving obstacle. The agent tries to make collisions with other objects during the learning process, and it does not take such actions after the learning.

5.2 Coordination of Multiple Behaviors

We consider three kinds of coordinations in which the previously learned behaviors are combined; simple summation of different action value functions, switching action value functions according to situations, and learning given the learned policies as *a priori* knowledge. The state spaces $\boldsymbol{S}^c$ for the coordinated behavior in these coordinations are a little bit different from each other according to their methods. To simplify the following explanations, let us consider to combine a goal-directed behavior ($Q^g(s^g, a)$) and a reflexive behavior ($Q^r(s^r, a)$) into a new one.

Basically, a state $s^c \in \boldsymbol{S}^c$ can be defined as a combined state of $\boldsymbol{S}^g$ and $\boldsymbol{S}^r$. We denote this combination as $\boldsymbol{S}^g \times \boldsymbol{S}^r$ or $(\boldsymbol{S}^g, \boldsymbol{S}^r)$. The number of $\boldsymbol{S}^c$ is theoretically a product of numbers of states of $\boldsymbol{S}^g$ and $\boldsymbol{S}^r$.

(a) Simple summation of different action value functions

The action value function of simple summation $Q^c_{ss}(s^c, a)$ for the coordinated behavior is given by;

$$Q^c_{ss}(s^c, a) = \max_{a \in \boldsymbol{A}} (Q^g((s^g, *), a) + Q^r((*, s^a), a)) \quad (3)$$

where $Q^g((s^g, *), a)$ and $Q^a((*, s^a), a)$ denote the extended action value functions for the goal-directed and reflexive behaviors in the new state space, respectively. $*$ means any states, therefore each of these functions considers only the original states and ignores the states of other behaviors. In this scheme, the selected action sometimes might not make any sense for both behaviors because the simple summation cannot consider combined new situations.

(b) Switching action value functions

The switching action value function $Q^c_{sw}(s^c, a)$ for the coordinated behavior is given by the following equation depending on a situation.

$$Q^c_{sw}(s^c, a) = \begin{cases} Q^r(s^r, a), & \text{in some situations} \\ Q^g(s^g, a), & \text{otherwise} \end{cases} \quad (4)$$

It seems hard to appropriately determine the situations to switch the functions $Q^g(s^g, a)$ and $Q^r(s^r, a)$. Simple situations we tried are the cases where only an opponent can be seen or where an opponent can be seen. In the former, the robot does not care about

collisions with the opponent when the ball or the goal can be observed, while in the latter the robot tries to avoid the opponent even if it is likely able to shoot a ball into the goal. Therefore, we need a carefully designed decision rule to switch the policies. The following method provides us with this rule by learning a new policy coping with new situations.

(c) Learning a new behavior

In the above methods, the previously learned action value functions are simply summed or switched. Therefore these methods ignore some situations inconsistent with the state spaces $\boldsymbol{S}^g$ or $\boldsymbol{S}^r$. Eventually, 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 (see [3,4] for more details).

A typical example is the case where a ball and the opponent are located at the same area and the ball is occluded by the opponent from the viewpoint of the robot. In this case, the robot cannot observe the ball, and therefore the corresponding state might be the state of "ball-lost," but it is not correct. Of course, if both the ball and the opponent can be observed, this situation can be considered consistent. This problem is resolved by adding new sub-states. In the above example, a new situation "occluded" is added, and the corresponding new sub-states are generated. In order to detect hidden states, we use χ^2-test.

5.3 Experiments

In addition to three kinds of coordination methods, we show the performance data by only using the policy Q^g which completely ignores the existence of the opponent. Table 1 shows the simulation result where the success rate of shooting per trial, the mean steps between collisions with the opponent, and the mean steps needed to get a shoot (success). In the case of only using Q^g, the robot tries to shoot a ball ignoring the opponent, and therefore it collides with the opponent many times and needs much more steps to get a shoot although the rate is as good as the learning method. The simple sum seems better in collision because avoiding behavior becomes dominant when the opponent approaches to it. However, it sometimes settles at one of the local maxima near the goal where shooting and avoiding behaviors are balanced, and therefore the shooting rate is the worst. The switching condition we set is to use shooting behavior unless only the opponent can be observed very largely. The robot got more shoots than the simple sum because it can avoid the local maxima. However, when it uses avoiding one, many actions not related to shooting behavior are chosen, and therefore it takes longest time step to get a shoot as a result. The learning method is the best in shooting rate, collision avoidance, and speed of shooting per trial.

Fig.8 shows a sequence of shooting behavior by the learning method. In these figures, the robot and the opponent are colored in black and gray, respectively. The lines emerged from them shows their visual angles. The opponent tries to chase after the robot with the probability of 50% as long as it can see the robot. Otherwise, it randomly moves.

Table 1 Simulation result

coordination method	success rate(%)	mean steps between collisions	mean steps to success
only Q^g	46.7	43.1	286.9
simple sum	33.2	77.5	231.2
switching	39.2	98.0	414.4
learning	46.7	238.1	128.3

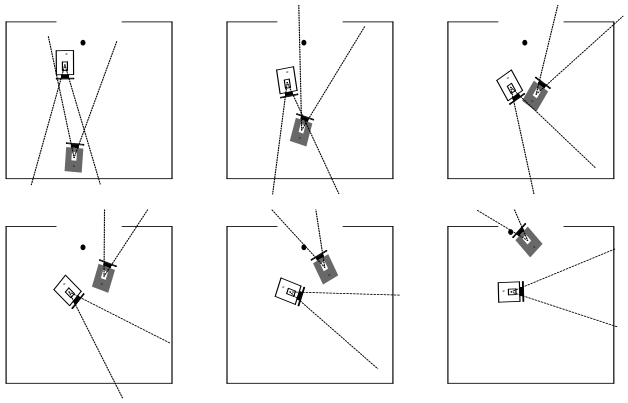

Fig.8 The robot succeeded in shooting a ball into a goal avoiding a moving keeper robot.

5.4 Efficient Learning by Scheduling Opponent Behaviors (Another LEM)

Next, we studied how the learning agent can improve its performance by the behavior of other agents. Intuitively, we can see the learning agent cannot learn at all if the opponent has the optimal policy to block the learner because of no success. Therefore, according to the basic idea of Learning from Easy Missions Paradigm (hereafter LEM) [3], we started with a stationary opponent (stationary obstacle), and then increase its velocity until the maximum one of the agent. **Fig.9** shows the succeeded shooting rate in terms of number of trials [1] with and without LEM. With LEM, the agent started learning in the environment with a stationary opponent, and then with a moving one of half speed (from the first arrow), and finally with one of the maximum speed (from the second arrow). While, without LEM, the agent starts from an opponent with the maximum speed, therefore the success ratio has not achieved the level with LEM. This figure tells that LEM seems essential for the learning from other competitive agents. **Fig.10** shows a sequence of images where the robot achieved the goal avoiding an opponent that is currently static.

[1] one trial ends when the agent succeeds in shooting or crosses over the field line

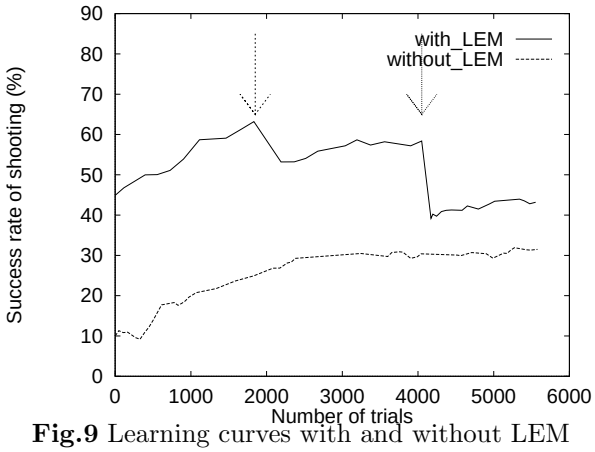

Fig.9 Learning curves with and without LEM

6 Discussion

We review the research issues involved in **RoboCup** with real robots. Realtime sensing capability is indispensable for quick motions of each player. Conventional methods of measurement and planning does not seem suitable. As we have shown, learning method seems encouraging although learning team plays such as passing and formation has not been attacked yet.

In this paper, we have defined the state-action space before learning. But, the state space construction problem is very difficult and closely related to "segmentation" problem, one of the most fundamental AI ones. We expect that **RoboCup** provides us a good test bed for this problem.

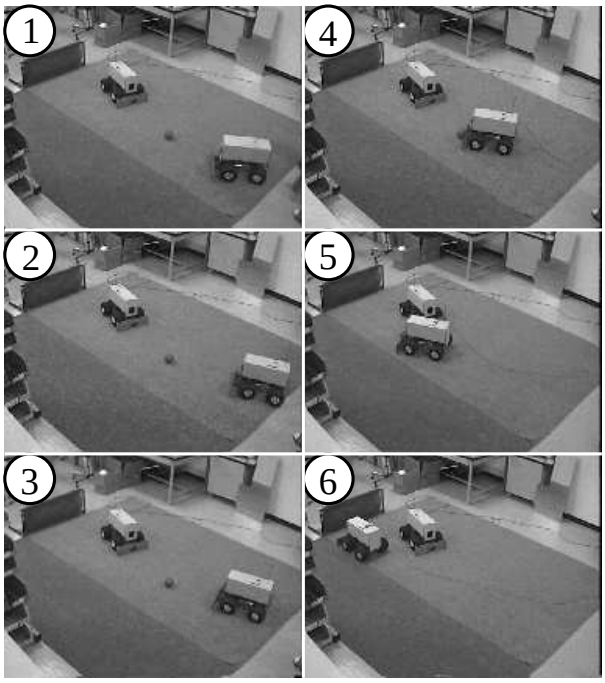

Fig.10 The robot succeeded in shooting a ball into a goal avoiding an opponent.

References

[1] M. Asada, S. Noda, and K. Hosoda. Action-Based Sensor Space Categorization for Robot Learning. In *Proc. of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

[2] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda. Vision-Based Reinforcement Learning for Purposive Behavior Acquisition. In *Proc. of IEEE International Conference on Robotics and Automation*, pp. 146–153, 1995.

[3] M. Asada, E. Uchibe, and K. Hosoda. Agents That Learn from Other Competitive Agents. In *Proc. of Machine Learning Conference Workshop on Agents That Learn from Other Agents*, 1995.

[4] 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.

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

[8] H. Kitano, M. Asada, Y. Kuniyoshi, and I. N. E. Osawa. Robocup : The Robot World Cup Initiative. In *Proc. of IJCAI-95 Workshop on Entertainment and AI/A-life*, 1995.

[9] Y. Takahashi, M. Asada, and K. Hosoda. Reasonable Performance in Less Learning Time by Real Robot Based on Incremental State Space SegmentationAction-Based Sensor Space Categorization. In *Proc. of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

[10] E. Uchibe, M. Asada, and K. Hosoda. Behavior Coordination for a Mobile Robot Using Modular Reinforcement Learning. In *Proc. of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

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