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

Minoru Asada
Dept. of Mechanical Engineering for Computer-Controlled Machinerey
Osaka University, Suita, Osaka 565 Japan
asada@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 [Asada *et al.*, 1994a; Asada *et al.*, 1995b], 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. As one extension of this first achievement, the method for autonomous state space construction is proposed [Asada *et al.*, 1995a]. In the second stage [Asada *et al.*, 1994b; Asada *et al.*, 1994c], 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 [Asada *et al.*, 1995c]. 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 [Connel and Mahadevan, 1993b]. In the reinforcement learning scheme, a robot and an environment are modeled by two synchronized finite state automatons interacting in a 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 [Sutton, 1992]. 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.

Some applications are recently reported to control robot arms [Saito and Fukuda, 1994] or mobile robots [Fagg *et al.*, 1994] in which the initial controller and the correct reward function are given in advance. Therefore, the robot learns the control policy given a great deal of knowledge about the environment and itself. We intend to apply the reinforcement learning algorithm to the task of purposive behavior acquisition in the real world with less knowledge about the environment and the robot.

Mahadevan and Connel [Connel and Mahadevan, 1993a] 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 [Kitano *et al.*, 1995]. 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. 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 [Asada *et al.*, 1994a], 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 (totally, 9 actions). As one extension of this first achievement, the method for autonomous state space construction was proposed [Asada *et al.*, 1995a]. In the second stage [Asada *et al.*, 1994b; Asada *et al.*, 1994c], 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 [Asada *et al.*, 1995c].

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 [Watkins, 1989]. We follow the explanation of Q-learning by Kaelbling [Kaelbling, 1993].

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. The world is modeled as a Markov process, making stochastic transitions based on its current state and the action taken by the robot. Let $T(s, a, s')$ be the probability of transition to the state s' from the current state-action pair (s, a). For each state-action pair (s, a), the *reward* $r(s, a)$ is defined.

The general reinforcement learning problem is typically stated as finding a policy [1] that maximizes the discounted sum of rewards received over time. This sum is 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. The value of γ is usually slightly less than 1.

Given definitions of the transition probabilities and the reward distribution, we can solve for the optimal policy, using methods from dynamic programming [Bellman, 1957]. A more interesting case occurs when we wish to simultaneously learn the dynamics of the world and construct the policy. Watkin's Q-learning algorithm gives us an elegant method for doing this.

Let $Q^*(s, a)$ be the expected return or *action-value function* for taking action a in a situation s and continuing thereafter with the optimal policy. It can be recursively defined as:

$$Q^*(s, a) = r(s, a) + \gamma \sum_{s' \in S} T(s, a, s') \max_{a' \in A} Q^*(s', a'). \tag{2}$$

Because we do not know T and r initially, we construct incremental estimates of the Q-values on-line. Starting with $Q(s, a)$ equal to an arbitrary value (usually 0), every time an action is taken, the Q-value is updated as follows:

$$Q(s, a) \Leftarrow (1 - \alpha)Q(s, a) + \alpha(r(s, a) + \gamma \max_{a' \in A} Q(s', a')). \tag{3}$$

where r is the actual reward value received for taking action a in a situation s, s' is the next state, and α is a learning rate (between 0 and 1).

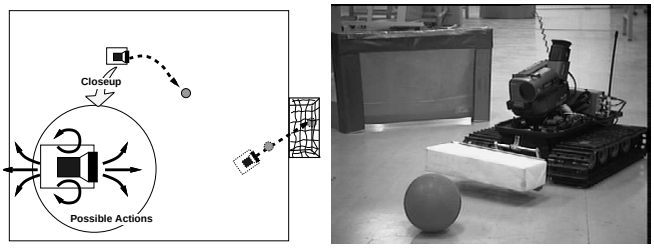
To speed up the learning time, we generate actions probabilistically based on Q values using a Boltzmann distribution. Given a situation s, we choose an action a with probability:

$$\frac{e^{Q(a,s)/T}}{\sum_{a \in \boldsymbol{A}} e^{Q(a,s)/T}} \tag{4}$$

This serves to make actions whose values are much better than the others be chosen with much greater likelihood. The *temperature* parameter T controls the amount of exploration (the degree to which actions other than the one with the best Q value are taken).

3 Shooting Behavior Acquisition [Asada *et al.*, 1995b]

3.1 Task, Assumptions, and a Real Robot System



(a) The task is to shoot a ball into a goal.

(b) A picture of the radio-controlled vehicle.

Figure 1: Task and our real robot.

The task for a mobile robot is to shoot a ball into a goal as shown in Figure 1(a). The problem we address here is how to develop a method which automatically acquires

[1] A policy f is a mapping from S to A.

strategies for doing this. We assume that the environment consists of a ball and a goal; the mobile robot has a single TV camera; and that the robot does not know the location/size of the goal, the size/weight of the ball, any camera parameters such as the focal length and tilt angle, or the kinematics/dynamics of itself. Figure 1(b) shows a picture of the real robot with a TV camera (Sony handy-cam TR-3) used in the experiments.

3.2 Construction of State and Action Spaces

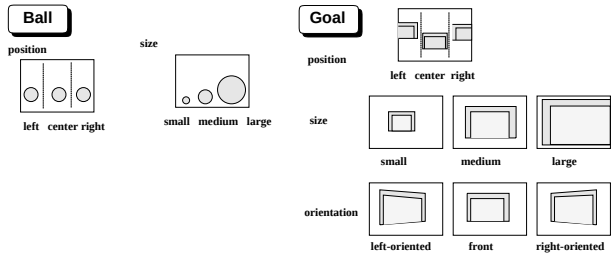


Figure 2: The ball sub-states and the goal sub-states

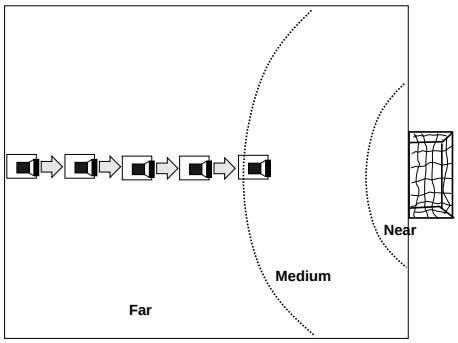


Figure 3: A state-action deviation problem

Figure 2 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"**: Figure 3 indicates this problem, the area representing the state "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 defined 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".

3.3 Learning from Easy Missions

In order to improve the learning rate, the whole task was separated into different parts in [Connel and Mahadevan, 1993a]. By contrast, we do not decompose the whole task into subtasks of finding, driblling, 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).

3.4 Action-Based State Space Construction [Asada *et al.*, 1995a]

In the previous section, first we divided the sensor space by hand, and then constructed the action space so that the sensor and action spaces can be consistent with each other. While, one can construct a state space fixing the action space first [Chapman and Kaelbling, 1991; Dubrawski and Reingnier, 1994]. Generally, the optimal state space design should be supported by the optimal action space design, and vice versa. This resembles the well-known "chicken and egg problem".

From a viewpoint of state space construction, the problem is twofold :

1. how to determine which information among the perceived data is necessary for the task, and
2. how to divide the selected parameter space into non-overlapping areas called "states" so that the robot can take an adequate action to perform the task.

Since the whole problem, that is, simultaneous construction of state and action spaces seems very difficult one to be solved, we propose a method for the second problem, and apply the method to the the same task. Basic ideas of our method are as follows:

- We construct a state space so that a group of sensation vectors in which an action command to achieve the goal is the same can be merged into one state even if these vectors appear to be different from each other.
- An action is defined as a sequence of the same action command in such a state. That is, we parameterized the length of action sequence in order to divide the sensor space.

3.5 Experimental results

Figure 4 shows a configuration of the real mobile robot system. 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. We constructed the radio control system of the vehicle [Inaba, 1993]. The image processing and the vehicle control system are operated by VxWorks OS on MC68040 CPUs which are connected with host

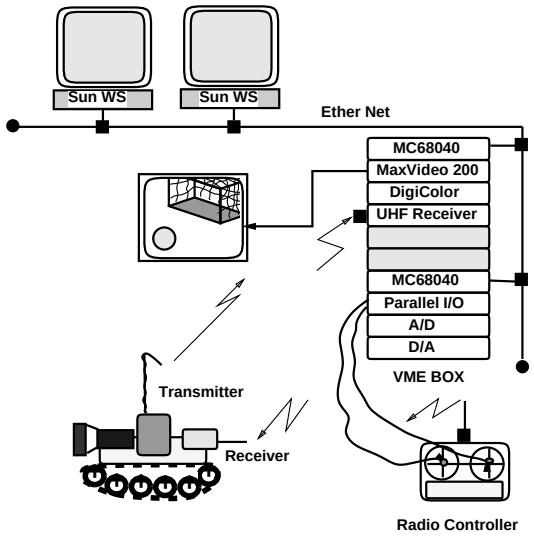


Figure 4: A configuration of the real system.

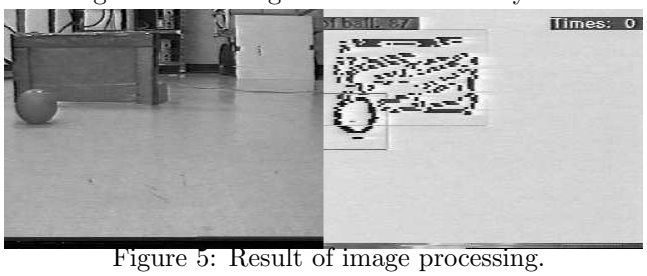


Figure 5: Result of image processing.

Sun workstations via Ether net. We have shown a picture of the real robot in Figure 1(b).

Figure 6 shows the projected map of the final result onto the ball-size and goal-size space when other parameters are all zeros. The intensity indicates the order of the division: the darker is the earlier. Grid lines indicate the boundaries divided by programmer in the previous work. The remainder of the state space in Figure 6 corresponds to infeasible situations such as "the goal is near and the ball is far" although we did not recognize such a meaningless state in the previous work. As we can see, the sensor space categorization by the proposed method (a set of ellipsoids) is quite different from the one designed by the programmer (rectangular grids) in the previous work.

Figure 7(a) shows how a real robot shoots a ball into a goal by using the state and action map obtained by the method. 16 images are shown in raster order from the top left to the bottom right in every 1.5 seconds, in which the robot tried to shoot a ball, but failed, then moved backward so as to find a position to shoot a ball, finally succeeded in shooting. Figure 7(b) shows a sequence of images taken by the robot during the task execution shown in Figure 7(a). Note that the backward motion for retry is just the result of learning and not hand-coded.

Table 1 compares the method with our previous work [Asada *et al.*, 1995b]. The search time in the previous work means the learning time in terms of the period of one action command (33ms) since the state space is given a priori. It takes about 500M (M=10^6) steps because the number of states is much larger. The proposed method performs better than the previous work. The size of the

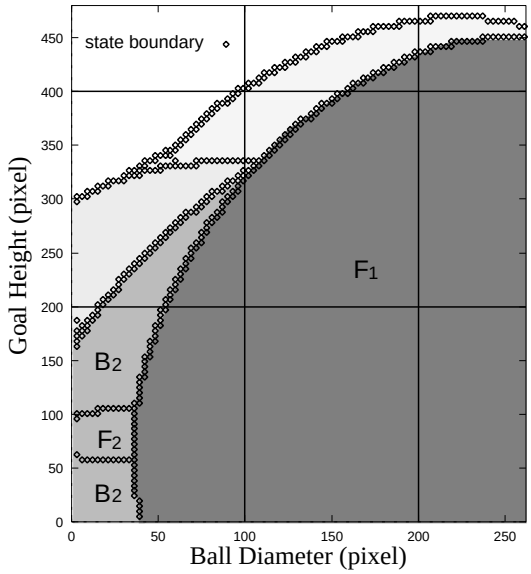


Figure 6: 2-D projection of the state space division

Table 1: *Comparison of the methods with our previous work*

	# of States	Search Time	Success Rate (%)
Previous work	243	500M*	77.4
Proposed method	33	41M	83.3

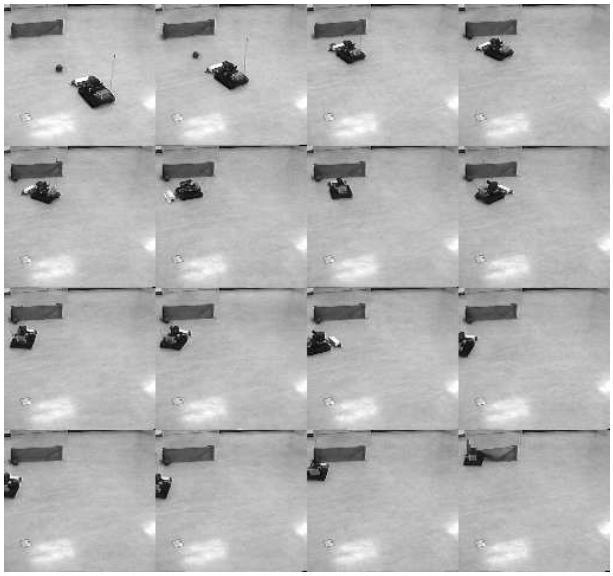
* indicates the Q-learning time.

state space is about 1/8 of that of the previous work. The search time is about 1/12 of the previous work.

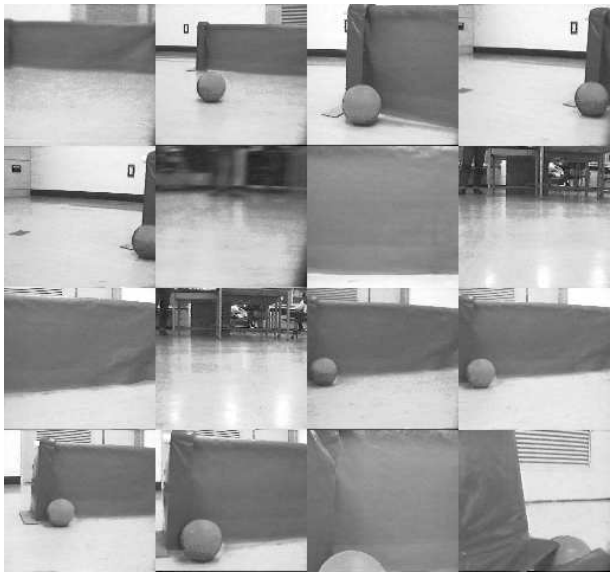
4 Shooting a Ball with Avoiding an Opponent [Asada *et al.*, 1994b; Asada *et al.*, 1994c; Asada *et al.*, 1995c]

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 Figure 8). 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 2. 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.

We assign a reward value 3 when the ball is entered into the goal or 0 otherwise for the shooting task, and -1 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 ($\gamma^g = 0.8$), and for the avoiding task $\gamma^r = 0.1$.



(a) The robot succeeded in finding and shooting a ball into the goal



(b) Images taken by the robot during the task execution

Figure 7:

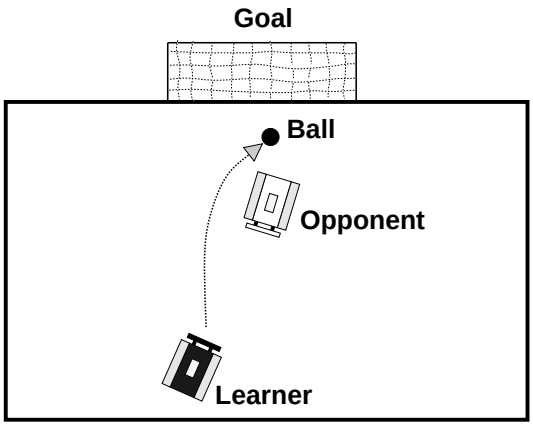


Figure 8: The task is to shoot a ball into the goal avoiding an opponent.

4.1 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.

4.2 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 explanatins, 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;

$$Q^c_{ss}(s^c, a) = \max_{a \in \boldsymbol{A}} (Q^g((s^g, *), a) + Q^r((*, s^a), a)) \qquad (5)$$

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.

$$Q^c_{sw}(s^c, a) = \begin{cases} Q^r(s^r, a), & \text{in some situations} \\ Q^g(s^g, a), & \text{otherwise} \end{cases} \quad (6)$$

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 [Asada *et al.*, 1994b; Asada *et al.*, 1994c] 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 substates. In the above example, a new situation "occluded" is added, and the corresponding new substates are generated.

The learning scheme is applied to both normal states and newly generated ones with different temperature parameters T in eqn(4) for the action selection in such a way that low temperature (conservative) is used around the normal states s^c and high temperature (random) around the new substates s^c_{sub} in order to reduce the learning time.

4.3 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 2 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.

Table 2: 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

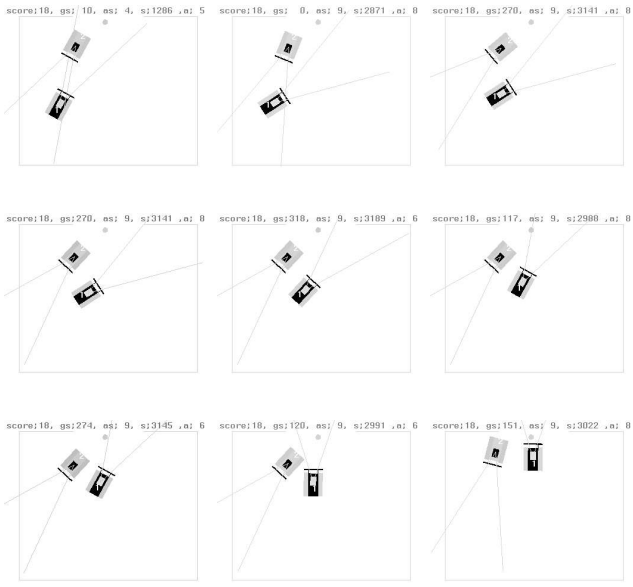


Figure 9: A shooting behavior of the learning method

Fig.9 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.

4.4 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) [Asada *et al.*, 1995b], we started with a stationary opponent (stationary obstacle), and then increase its velocity until the maximum one of the agent. Figure 10 shows the succeeded shooting rate in terms of number of trials [2] with and without LEM. With LEM, the agent started learning with a stationary opponent, and then with 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.11** shows a sequence of images where the robot achieved the goal avoiding an enemy that is currently static.

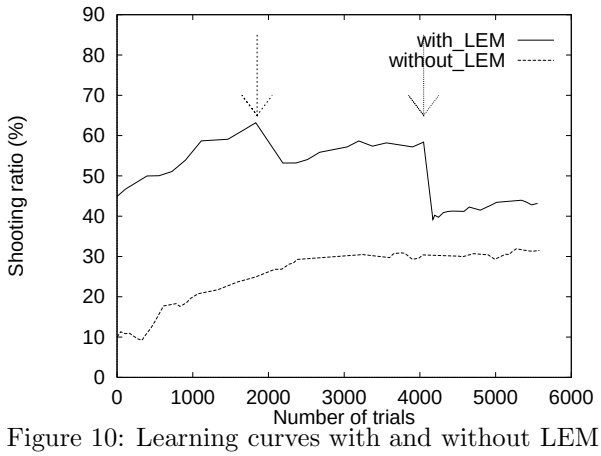


Figure 10: Learning curves with and without LEM

5 Discussion

We review the research issues involved in **RoboCup** with real robots. In order to build a team of robots, each robot should be compact and simple, therefore implementation of only "pushing behavior" would be a reasonable strategy. Currently, realization of a humanoid type robot seems only for the demonstration track, and to the best of our knowledge, Prof. Inaba, University of Tokyo, is challenging this problem.

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.

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


Figure 11: The robot succeeded in shooting a ball into a goal avoiding an opponent.

The state space construction problem is closely related to "segmentation" problem, one of the most fundamental AI ones. We expect that our problem formulation of the state space construction based on action could project a light to this difficult problem becasue we believe that segmentation of sensory information and action cannot be performed without any physiacl interactions with the environment, and **RoboCup** provides us a good test bed for this problem.

Acknowledgements

The research topics described here are supported by the Japanese Sience Research Program under the Project Numbers 06650301, 07455112, and 07243214 and of the Grand-in-Aid for sientific research from the Ministry of Education, Sience, and Culture.

The author thanks Koh Hosoda, Shoichi Noda, Eiji Uchibe, and Sukoya Tawaratsumida for their constructive discussions and invaluable efforts to realize the work described in this paper.

References

[Asada *et al.*, 1994a] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda. "purposive behavior acquisition on a real robot by vision-based reinforcement learning". In *Proc. of MLC-COLT (Machine Learning Conference and Computer Learning Theory) Workshop on Robot Learning*, pages 1–9, 1994.

[Asada *et al.*, 1994b] M. Asada, E. Uchibe, S. Noda, S. Tawaratsumida, and K. Hosoda. "A vision-based reinforcement learning for coordination of soccer playing behaviors". In *Proc. of AAAI-94 Workshop on AI and A-life and Entertainment*, pages 16–21, 1994.

[Asada *et al.*, 1994c] M. Asada, E. Uchibe, S. Noda, S. Tawaratsumida, and K. Hosoda. "coordination of multiple behaviors acquired by vision-based reinforce-

ment learning". In *Proc. of IEEE/RSJ/GI International Conference on Intelligent Robots and Systems 1994 (IROS '94)*, pages 917–924, 1994.

[Asada *et al.*, 1995a] M. Asada, S. Noda, and K. Hosoda. Non-physical intervention in robot learning based on lfe method. In *Proc. of Machine Learning Conferen Workshop on Learning from Examples vs. Programming by Demonstration*, pages 25–31, 1995.

[Asada *et al.*, 1995b] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda. Vision-based reinforcement learning for purposive behavior acquisition. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, pages 146–153, 1995.

[Asada *et al.*, 1995c] M. Asada, E. Uchibe, and K. Hosoda. Agents that learn from other competitive agents. In *Proc. of Machine Learning Conferen Workshop on Agents That Learn from Other Agents*, pages 1–7, 1995.

[Bellman, 1957] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[Chapman and Kaelbling, 1991] D. Chapman and L. P. Kaelbling. "Input generalization in delayed reinforcement learning: An alogorithm and performance comparisons". In *Proc. of IJCAI-91*, pages 726–731, 1991.

[Connel and Mahadevan, 1993a]
J. H. Connel and S. Mahadevan. "Rapid task learning for real robot". In J. H. Connel and S. Mahadevan, editors, *Robot Learning*, chapter 5. Kluwer Academic Publishers, 1993.

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

[Dubrawski and Reingnier, 1994]
A. Dubrawski and P. Reingnier. Learning to categorize perceptual space of a mobile robot using fuzzy-art neural network. In *Proc. of IEEE/RSJ/GI International Conference on Intelligent Robots and Systems 1994 (IROS '94)*, pages 1272–1277, 1994.

[Fagg *et al.*, 1994] A. H. Fagg, D. Lotspeich, and G. A. Bekey. "A reinforcement learning approach to reactive control policy design for autonomous robots". In *Proc. of 1994 IEEE Int. Conf. on Robotics and Automation*, pages 39–44, 1994.

[Inaba, 1993] M. Inaba. "Remote-brained robotics: Interfacing ai with real world behaviors". In *Preprints of ISRR'93*, Pitsuburg, 1993.

[Kaelbling, 1993] L. P. Kaelbling. "Learning to achieve goals". In *Proc. of IJCAI-93*, pages 1094–1098, 1993.

[Kitano *et al.*, 1995] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. "robocup: The robot world cup initiative". In *Proc. of IJCAI-95 Workshop on Entertainment and AI/A-life*, 1995.

[Saito and Fukuda, 1994] F. Saito and T. Fukuda. "Learning architecture for real robot systems – extension of connectionist q-learning for continuous robot control domain". In *Proc. of 1994 IEEE Int. Conf. on Robotics and Automation*, pages 27–32, 1994.

[Sutton, 1992] R. S. Sutton. "Special issue on reinforcement learning". In R. S. Sutton(Guest), editor, *Machine Learning*, volume 8, pages –. Kluwer Academic Publishers, 1992.

[Watkins, 1989] C. J. C. H. Watkins. *Learning from delayed rewards"*. PhD thesis, King's College, University of Cambridge, May 1989.