Agents That Learn from Other Competitive Agents

Minoru Asada, Eiji Uchibe, and Koh Hosoda
Dept. of Mech. Eng. for Computer-Controlled Machinery
Osaka University, 2-1, Yamadaoka, 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. In the first stage [Asada *et al.*, 1994a], a robot learned to shoot a ball into a goal. 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. This can be considered as a problem of learning from other agents. The big difference from the existing schemes such as learning by demonstration [Kuniyoshi and Inoue, 1993] and social learning [Mataric, 1994] is that the other agents is not always friendly nor suggestive to help the agent learn. Rather, the other agents are involved in the task which the agent has to accomplish. That is, they are competitive agents. In this paper, we discuss several issues on the problem of "learning from other competitive agents" we are facing with in our project.

1 INTRODUCTION

Building a robot that learns to perform a task has been acknowledged as one of the major challenges facing Robotics and AI. 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, 1993]. 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.

We have been doing a research on vision-based reinforcement learning and applied the method to build real soccer playing robots. In the first stage [Asada *et al.*, 1994a], we constructed 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). 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 (See Figure 1). This can be considered as a problem of learning from other agents. The big difference from the existing schemes such as learning by watching [Kuniyoshi and Inoue, 1993] and social learning [Mataric, 1994] is that the other agents is not always friendly nor suggestive to help the agent learn. Rather, the other agents are involved in the task which the agent has to accomplish. That is, they are competitive agents. In this paper, we discuss several issues on the problem of "learning from other competitive agents" we are facing with in our project.

2 CLASSIFICATION AND SCOPE

There might be several ways to categories a methodology of "Learning from Other Agents" depending on who are other agents. Colleagues, opponents, teachers, critics, or gods. From a viewpoint of Robot Learning, we classify the methodology in terms of character and involvement of other agents in the context of learning strategy. Cooperative or competitive? Are they included in the environment or excluded from it? Table 1 shows this classification.

The most popular method of "learning from other agents" is Learning by Demonstration by Kuniyoshi et al. [Kuniyoshi and Inoue, 1993] in which other agents

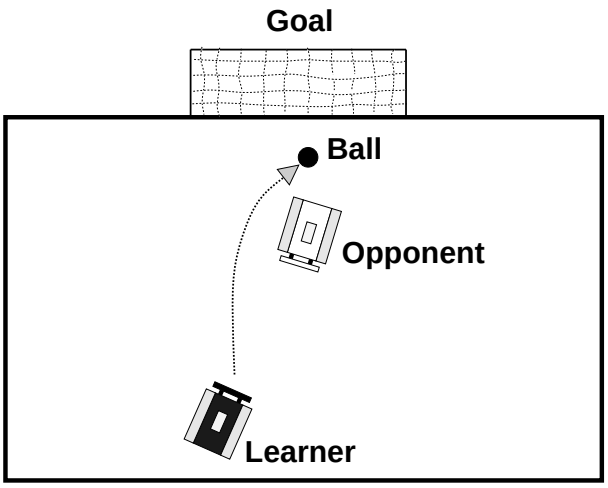


Figure 1: The task is to shoot a ball into the goal avoiding an opponent.

	cooperative	competitive
excluded from environment	Learning by Demonstration [Kuniyoshi and Inoue, 1993] Learning from External Critic [Whitehead, 1991]	?
included in environment	social learning [Mataric, 1994]	competition [Littman, 1994] [Asada et al., 1994b] [Asada et al., 1994c]

Table 1: Classification of the methodology

are kind and suggestive human operators. Learning from External Critic (hereafter LEC) [Whitehead, 1991] is also categorized into this class although an critic (other agent) has not shown any demonstrations but gives an advice each time an agent takes an action. In these cases, other agents are cooperative but not involved in the environment in which an agent learns to to accomplish a given task. While, in the method of group learning or social learning [Mataric, 1994], cooperative agents are involved in the environment and therefore other agents can be regarded as a part of the environment for a learning agent.

Other agents involved in the environment are not always cooperative. Some of them might be competitive with the learning agent in real robot environments. Littman [Littman, 1994] proposed Markov game as a framework for multi-agent reinforcement learning in which one to one competition (a soccer-like game) was dealt. Both agents try to shoot a ball into the opponent's goal and at the same time to block an opponent to shoot a ball into its own goal. Asada et al. [Asada *et al.*, 1994b; Asada *et al.*, 1994c] proposed a method of behavior coordination for the similar task to Littman's, but the task is to shoot a ball into a goal avoiding an opponent (a goal keeper) and blocking behavior is not included. In these two cases, other agents are competitive and involved in the environment.

In the current state, we cannot find any sense in the learning from other competitive agents who are excluded from the environment ("?" in the table).

In this paper, we discuss the issues in the learning from other competitive agents method, intending to apply the method to real robot applications such as real robot soccer playing (ex., [Asada *et al.*, 1994a]).

The remainder of this article is structured as follows: In the next section, we give a brief overview of the Q learning and three kinds of coordinations of multiple behaviors. We then explain the task, and how to construct state and action spaces in our method. Next, we show the experiments with computer simulations and on-going real robot system. Finally, we give discussions.

3 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.

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

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:

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

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:

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

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:

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

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).

4 CONSTRUCTING STATE AND ACTION SPACES

In order to apply the Q-learning scheme to each of two subtasks, we define a number of sets and parameters for each of them. The existing applications of the reinforcement learning have constructed the state and action spaces in such a way that each action causes the state transition (ex. one action is forward, backward, left, or right, and states are encoded by the locations (coordinates) of the agent) in order to make the quantization problem (the structural credit assignment problem) easy. This makes a gap between the computer simulations and real robot systems. Each space should reflect the corresponding physical space in which a state or an action can be perceived or taken. Then, we construct these spaces considering the sensor resolution and control parameter resolution for the actuator. Figure 2 shows sub-states used in each subtask. For the shooting task, ball and goal sub-states are used ($\boldsymbol{S}^g$), and for the avoiding behavior, opponent sub-state is used ($\boldsymbol{S}^r$).

The robot moves around the field by a PWS (Power Wheeled Steering) system with two independent motors. Since we can send the motor control command

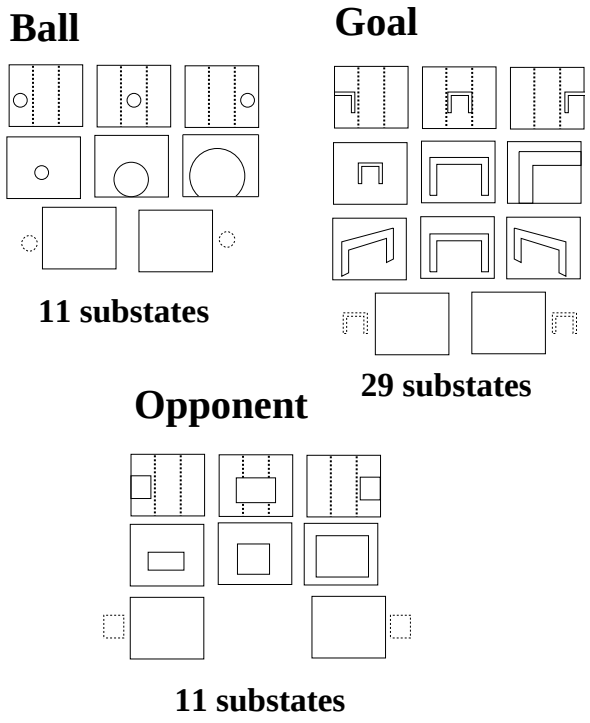


Figure 2: Sub-states for both tasks

to each of two motors independently, we quantized the action set in terms of two motor commands ω_l and ω_r, each of which has 3 sub-actions (forward, stop, and backward motions, respectively). Totally, we have 9 actions in the action set $\boldsymbol{A}$.

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 $\gamma^r = 0.1$.

5 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, and such actions should be learned by using the following update equations instead of eqns.

$$Q^r(s,a) \Leftarrow (1-\alpha)Q^r(s,a) + \alpha(r + \gamma^a \min_{a' \in A} Q^r(s',a')). \quad (5)$$

$$f^r(s) \leftarrow a \quad \text{such that} \quad (6)$$

$$Q^r(s,a) = \min_{b \in \boldsymbol{A}} Q^r(s,b). \quad (7)$$

After learning, the decision of action selection is done based on eqn.(7). That is, the agent tries to make collisions with other objects during the learning process, and it does not take such actions after the learning. We studied the following situations:

1. **Stationary Obstacles:** This is the simplest case for the agent to avoid. What the agent has to learn is the relationship between the action commands and its effects in the environment. To do that the state space is constructed in terms of the size and position of the other agent. The learning has converged quickly and the agent could avoid the other agent adequately as long as the agent can observe the other agent.
2. **Moving Agents:** The next step is that the other agent is not stationary but moving around. This case is also separated into two cases according to the property of motions, that is, random motion or intended motion such as approaching toward the agent. In the case of random motions, the learning has not converged because the policy cannot be constructed due to randomness of other agent motions. While, if the other agent has a fixed policy to behave such as an intention to block the learner, the learning has converged. Then, we studied other cases in which the other agent shows random motions or the fixed policy actions stochastically. According to the ratio of the random motion, the learning time changed or the learning has not converged. If the other agent shows block motions against the learner with the ratio of 50% as long as it can observe the learner otherwise random motions, the learning has converged but the policy shows that the difference of the action values between the optimal action (collision) and other actions is not so large. To improve this situation, we add the changes in size and position of other agent in image captured by the learner into the state space. Although the size of the state space has increased, the learner could behave adequately whatever actions the other agent shows.

6 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;

$$Q^c_{ss}(s^c,a) = \max_{a \in \boldsymbol{A}}(Q^g((s^g,*),a) + Q^r((*,s^a),a)) \quad (8)$$

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.

$$Q^c_{sw}(s^c,a) = \begin{cases} Q^r(s^r,a), & \text{in some situations} \\ Q^g(s^g,a), & \text{otherwise} \end{cases} \quad (9)$$

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. The method is as follows;

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

$$\begin{array}{lcl} Q^c_{rl}(s^c, a) & = & Q^c_{ss}(s^c, a) \\ Q^c_{rl}(s^c_{sub}, a) & = & \text{original value of } Q^c_{ss}(s^c, a) \end{array} \quad (10)$$

   (b) control the temperature parameter 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 rate.

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 $s^g \in \boldsymbol{S}^g$ 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 $s^c_{sub} \in \boldsymbol{S}^c$. 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.

6.1 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

Fig.3 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. In this case, we did not add the changes in size and position of the opponent into the state space because it causes too huge state space to learn within reasonable time.

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.*, 1995], we started with a stationary opponent (stationary obstacle), and then increase its velocity until the maximum

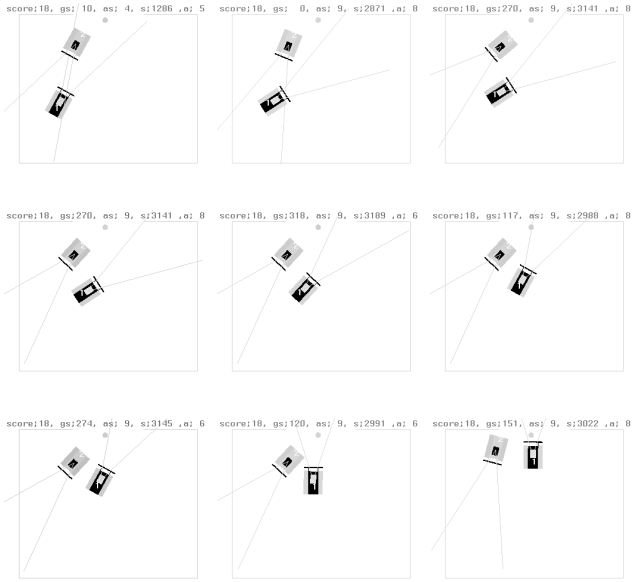


Figure 3: A shooting behavior of the learning method

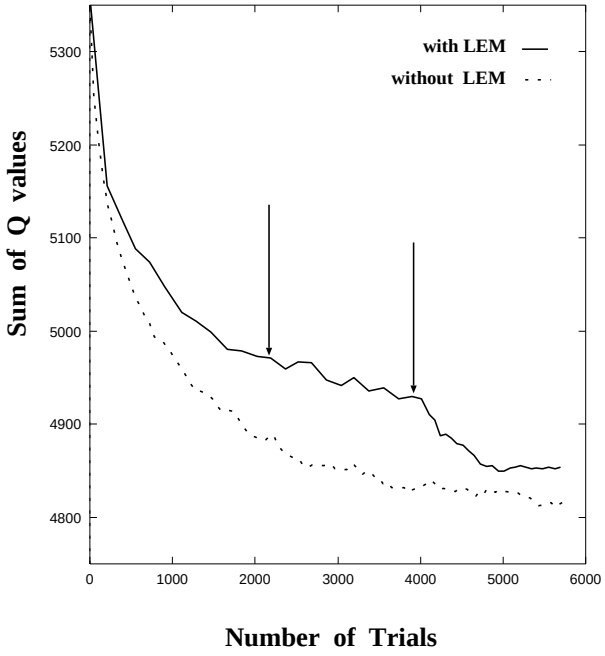


Figure 4: Learning curves with and without LEM

one of the agent. Figure 4 shows the learning curve (summation of maximum Q-values in terms of action) in terms of number of trials [2] with and without LEM. Since the initial value is overestimated due to simple summation of two action values, the learning curve starts from the high value. 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 summation of Q-values drastically decreased, and has not converged to the level with LEM. This figure tells that LEM seems essential for the learning from other competitive agents.

Fig.5 shows a picture of the real robot with a TV camera (Sony handy-cam TR-3) and video transmitter.

Fig.6 shows the result of the image processing where a ball (front left), a goal (center-back), and an enemy (right) are detected and their positions are calculated in real time (1/30 seconds). **Fig.7** shows a sequence of images where the robot achieved the goal avoiding an enemy that is currently static.

7 DISCUSSION

We have shown one aspect of the learning methods from other agents. In our experiments, we have not made the opponent learn because the learning seems hard unless the opponent policy has been fixed. If the state space is very small, multi-agent learning might be possible [Littman, 1994]. However, in applying the learning method in real robot applications, we need to reduce the huge search space by controlling other agents' behaviors. From this sense, we have not dealt with complete competitive agents.

Intuitively, simultaneous learning of two competitive agents seems very difficult because the policy to be learned cannot obtained from other agent of which policy has not been fixed. If it is possible [Littman, 1994], there should be some constraints. Otherwise, it cannot be justified. We need more study on this topic.

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.

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

Acknowledgements

This research was supported by the Japanese Sience Research Program under the Project Number 07455112 of the Grand-in-Aid for sientific research from the Ministry of Education, Sience, and Culture.

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*

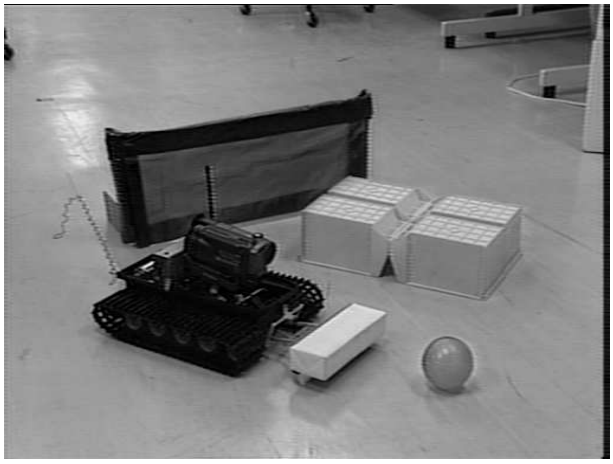
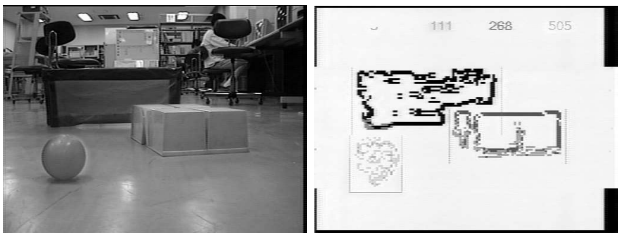


Figure 5: A picture of the radio-controlled vehicle near the goal with a ball and an opponent (a group of four boxes).



(a) input image (b) detected image

Figure 6: Detection of a ball (front left), a goal (center back) and an enemy (right).

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

[Asada *et al.*, 1995] 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.

Figure 7: The robot succeeded in shooting a ball into a goal avoiding an opponent.

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

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

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

[Kuniyoshi and Inoue, 1993] Y. Kuniyoshi and H. Inoue. Qualitative recognition of ongoing human action sequence. In *Proc. of IJCAI-93*, pages 1600–1609, 1993.

[Littman, 1994] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proc. of Conf. on Machine Learning-1994*, pages 157–163, 1994.

[Mataric, 1994] M. J. Mataric. Learning to behave socially. In *Proc. of the 3rd Int. Conf. on Simulation and Adaptive Behaviors – From animals to animats 3 –*, pages 453–462, 1994.

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

[Whitehead, 1991] S. D. Whitehead. "A complexity analysis of cooperative mechanisms in reinforcement learning". In *Proc. AAAI-91*, pages 607–613, 1991.