Vision-Based Behavior Acquisition For A Shooting Robot By Using A Reinforcement Learning

Minoru Asada, Shoichi Noda, Sukoya Tawaratsumida, 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 propose a method which acquires a purposive behavior for a mobile robot to shoot a ball into the goal by using a vision-based reinforcement learning. A mobile robot (an agent) does not need to know any parameters of the 3-D environment or its kinematics/dynamics. Information about the changes of the environment is only the image captured from a single TV camera mounted on the robot. An action-value function in terms of state is to be learned. Image positions of a ball and a goal are used as a state variable which shows the effect of an action previously taken. After the learning process, the robot tries to carry a ball near the goal and to shoot it. Both computer simulation and real robot experiments are shown, and discussion on the role of vision in the context of the vision-based reinforcement learning is given.

1 Introduction

Due to its globally perceptive capability, vision seems indispensable for autonomous agent(s) to acquire reactive and purposive behaviors that can be obtained through interactions with its environment. The existing deliberative and incremental approaches in computer vision, however, do not seem to have made the great advances in this context because these methods often need huge amount of computation time which is fatal in real time execution of robot tasks, and they offer general descriptions of the scene which might need more time to be transformed into the specified descriptions needed to accomplish tasks at hand. Rather, these general descriptions are hard to be properly evaluated unless the task or purpose of an agent is specified. From this viewpoint, purposive, task-oriented, or so-called *behavior-based approach* seems promising to evaluate the role of vision (when, where, and what kind of information is necessary and how accurate they should be) and finally to realize autonomous agent.

Since Brooks [1, 2] proposed the behavior-based approach, his group invented several kinds of behavior-based robots, such as [3] and [4]. Although these robots can take reflexive actions against the environment, we still lack a capability of *generating purposive behaviors each of which consists of a sequence of actions to achieve the goal*. We are studying the feasibility of providing this capability with our robot by using vision. Arkin [5] proposed a hybrid approach with reactive and deliberative methods for navigation task. He encoded *a priori* world knowledge into the motor schemas in order to generate the purposive behaviors. Here, we intend to solve this problem with less world knowledge and expect our robot to learn a behavior though interactions with dynamic environment.

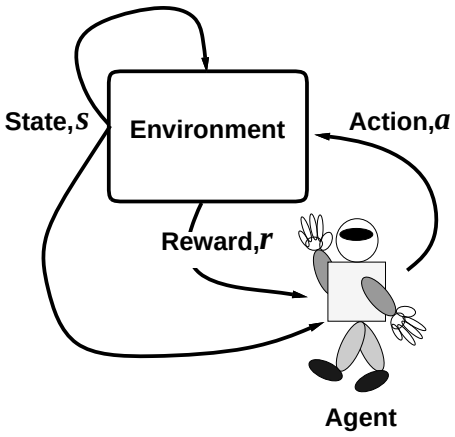


Figure 1: **The basic model of robot-environment interaction**

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]. Fig.1

shows the basic model of robot-environment interaction, where a robot and 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 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 determined by the extent to which it can be scaled to larger and complex robot learning tasks. Many theoretical works have argued on the convergence time of the learning, and how to speed up it by using some heuristics and to extend these techniques from a single goal task to multiple ones [7]. However, almost of them have only shown computer simulations, and only a few real robot applications are reported which are simple and less dynamic [8, 9]. Especially, the use of vision in the reinforcement learning is very rare.

To the best of our knowledge, only Whitehead and Ballard [10] argued this problem. Their task is a simple manipulation of blocks on the conveyer belt. Although each block is colored to be easily discriminated, they still have a large size of state space. To cope with this problem, they assumed that observer could control its gaze to attended object so as to reduce the size of the state space. However, this causes so-called "perceptual aliasing" problem. That is, both the observer motion and actual changes happened in the environment cause the changes inside the image captured by the observer. Therefore, it seems difficult to discriminate the both from only the image. Then, they proposed a method to cope with this problem by adopting the internal states and separating action commands into "Action frame" and "Attention frame" commands. Thus, they encoded encoded *a priori* world knowledge into state and action spaces.

In order to make the role of the reinforcement learning evident in realizing autonomous agents, we need more applications in more dynamic and complex environment. In this paper, we propose a method which acquires a purposive behavior for a mobile robot to shoot a ball into the goal by using a vision-based reinforcement learning. We apply Q-learning method [11], one of the widely used reinforcement learning schemes to our problem. The robot is expected to learn a shooting behavior without world knowledge such as 3-D locations and sizes of the goal and the ball or the kinematics and dynamics of the robot itself. All the information the robot can capture is the image positions of the ball and the goal from which we can infer changes in the world caused by the robot's actions.

The remainder of this article is structured as follows: In the next section, we give a brief overview of Q-learning. We then explain the task and the learning scheme in our method. Next, we show the experiments with computer simulations and a real robot system. Finally, we give concluding discussions.

2 Q-learning

Before getting into the details of our system, we briefly review the basics of Q-learning. For more through treatment, see [12]. We follow the explanation of Q-learning by Kaelbling [13].

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 that the world will transit to the next 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 that maximizes discounted sum of the reward received over time. A policy f is mapping from $\boldsymbol{S}$ to $\boldsymbol{A}$. 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 and is just slightly less than 1.

Given definitions of the transition probabilities and the reward distribution, we can solve the optimal policy, using methods from dynamic programming [14]. 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 $Q(s,a)$ at any value, usually 0, every time an action is taken update the Q value 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 leaning rate (between 0 and 1). The following is a simple version of the 1-step Q-learning algorithm we used here.

Initialization: $Q \leftarrow$ a set of initial values for the action-value function (e.g., all zeros).
Repeat forever:

1. $s \leftarrow$ the current state
2. Select an action a that is usually consistent with the policy f but occasionally an alternate.
3. Execute action a, and let s' and r be the next state and the reward received, respectively.
4. Update $Q(s,a)$:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a' \in A} Q(s',a')). \tag{4}$$

5. Update the policy f:

$$f(s) \leftarrow a \quad \text{such} \quad \text{that} \quad Q(s,a) = \max_{b \in \boldsymbol{A}} Q(s,b) \tag{5}$$

3 Task and Assumptions

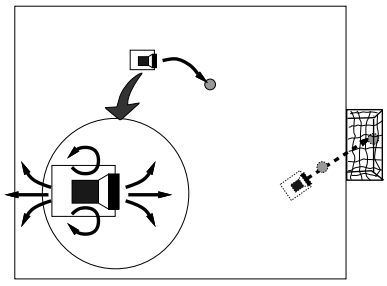


Figure 2: **The task is to shoot a ball into the goal.**

The task for a mobile robot is to shoot a ball into the goal as shown in **Fig.2**. The problem we are attacking here is to develop a method which automatically acquires strategies how to do this. As a first step, we simplify the environment in such a way that the environment consists of only a ball the robot can kick and a goal fixed on the ground.

If we know the exact three-dimensional parameters of the environment and kinematics and dynamics of the robot, we might be able to develop several methods to control it to shoot a ball into a goal. This is not our intention. We intend to start with the visual information only, that is, the image positions of the ball and the goal. That is all the robot captures from the environment. In order for the robot to take an action against the environment, it has several motion commands such as forward and turn left (See **Fig.2**). Note that the robot does not even know any physical meanings for these motion commands. The effects of an action against the environment can be informed to the robot only through the visual information. To enable to do that, the robot has to track the ball and/or the goal inside image continuously. **Fig.3** shows a mobile robot, a ball, and a goal we used in the real experiments.

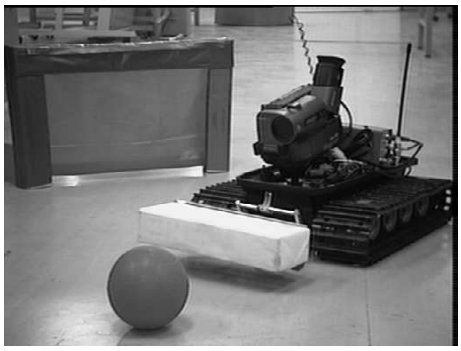


Figure 3: **A picture of the radio-controlled vehicle.**

4 Construction of State and Action Sets

In order to apply Q-learning scheme to the task, we define a number of sets and parameters. Many existing applications of the reinforcement learning schemes have constructed the state and action spaces in such a way that each action causes a state transition (e.g. one action is forward, backward, left, or right, and states are encoded by the locations 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 (an action) can be perceived (taken). However, such construction of state and action spaces sometimes causes a "state-action deviation" problem. In the followings, we describe how

to construct the state and action spaces, and then how to cope with the state-action deviation problem.

4.1 Construction of Each Space

(a) a state set S

Only the information the robot can obtain about the environment is the image supposed to be capturing the ball and/or the goal. The ball image is quantized into 9 sub-states, combinations of three positions (left, center, and right) and three sizes (large (near), medium, and small (far)). The goal image has 27 sub-states, combinations of three parameters each of which is quantized to three levels. Each sub-state corresponds to one posture of the robot toward the goal, that is, position and orientation of the robot in the field. In addition to these 243 (27×9) states, we add other states such as these cases in which only the ball or the goal is captured in the image.

After some simulations, we realized that as long as the robot captures the ball and the goal positions in the image, it succeeds in shooting a ball. However, once it lost the ball, it randomly moves because it does not know to which direction it should move to find the ball. This happens because the ball-lost state is just one, therefore it cannot determine in which direction the ball is lost. Then, we separate the ball-lost state into two states; the ball-lost-into-right and the ball-lost-into-left states. Also, we set up goal-lost-into-right and goal-lost-into-left states. This made the robot behavior much better. As a result, we totally have 319 states in the set $\boldsymbol{S}$.

(b) an action set A

The robot can select an action to be taken against the environment. In real system, 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 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 back motions, respectively). Totally, we have 9 actions in the action set $\boldsymbol{A}$.

Due to the peculiarity of the visual information, that is, a small change near the observer might cause a large change in image and vice versa, causes a state-action deviation problem because each action produces almost the same amount of motion in the environment. In our case, the resolution of robot action is much higher than that of state space. Therefore, the robot might frequently transits to the same state. This is highly undesirable because the variance of the state transitions is vary large, and therefore the learning does not converged correctly. Then, we reconstruct the action space as follows. Each action defined in the above is called an action primitive. The robot continues to take one action primitive until the current state changes. This sequence of the action primitive is called an action. The number of action primitives needed for state changes has no meanings. Once the state has changed, we apply the update equation (3) of the action value function.

(c) a reward and a discounting factor γ

We assign a reward value 1 when the ball was entered into the goal or 0 otherwise. This makes the learning very time-consuming. Although adopting a reward function in terms of distance to the goal state makes the learning time much shorter, it seems difficult to avoid the local maxima of the action-value function Q.

A discounting factor γ is used to control to what degree rewards in the distant future affect the total value of a policy. In our case, we set the value at slightly less than 1 ($\gamma = 0.8$).

5 Experiments

The experiment consists of two parts: first, learning the optimal policy f through the computer simulation, then apply the learned policy to a real situation. The merit of the computer simulation is not only to check the validity of the algorithm but also to save the running cost of the real robot during the learning process. Still real experiments are necessary because the computer simulation cannot completely simulate the real world [15].

5.1 Simulation

We performed the computer simulation with the following specifications (the unit is an arbitrary-scaled length). The field is a square of which side length is 200. The goal post is located at the center of the side line of the square (see Fig.2) and its height and width are 10 and 50, respectively. The robot is 16 wide and 20 long and kicks a ball of diameter 6. The camera is horizontally mounted on the robot (no tilt), and its visual angle is 36 degrees. The mass of the ball is negligible compared to that of the robot. Other parameters such as a bounding factor between the ball and the robot, viscous friction between the ball and the field and so on are properly chosen to simulate the real world.

First, we place the ball and the robot at arbitrary positions. In almost all cases, the robot crossed over the field line without shooting a ball into the goal.

This means that the learning has not converged after many trials (three days running on SGI Elan with R4000). This situation resembles a case in which the small child tries to shoot a ball into the goal, but he (or she) cannot imagine which direction and how far the goal is because a reward is received only after the ball has entered into the goal. Further, he (or she) does not know how to choose an action from several action commands. This is the famous *delayed reinforcement* problem due to no explicit teacher signal that indicates the correct output at each time step. Then, we construct the learning schedule such that the robot can learn in easy situations at early stages and learn in more difficult situations at later stages.

We began the learning of the shooting behavior by setting the ball and the robot near the goal. Once the robot succeeds in a shooting, the robot begins to learn (the sum of Q increases), but after that the robot wonders again in the field. After many iterations of these successes and failures, the robot learned to shoot a ball into the goal when the ball is near the goal. After that, we place the ball and the robot slightly further from the goal, and repeat the robot learning again.

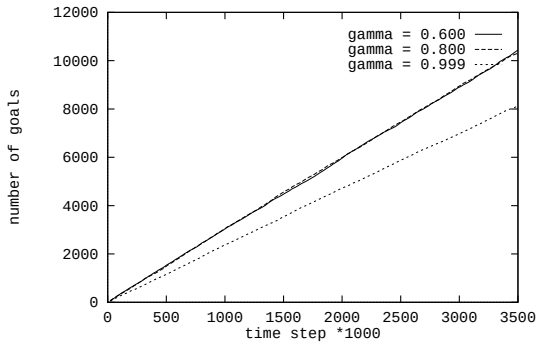
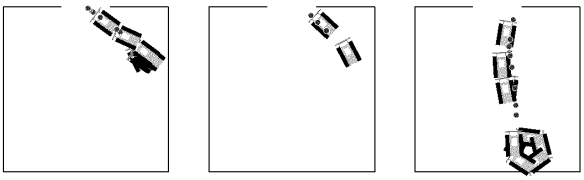


Figure 4: **Number of goals in terms of γ**

Fig.4 show the accumulated number of shooting goals in terms of the temporal discount factor γ, where the number of goals with larger γ (0.999) is lower than that with smaller ones (0.6 and 0.8). The reason is as follows. When the temporal discount factor γ is very close to 1 (almost no discount), the reward received after the goal is almost the same whichever path is selected. While, if γ is small, the robot try to take a shorter path which means more rewards. However, for a too small γ, the robot loses the way to the goal. **Fig.5** shows some kinds of behaviors during the learning process. (a) and (b) show the difference between the shooting behaviors with different γs. In (a),



(a) shooting ($\gamma = 0.999$) (b) shooting ($\gamma = 0.6$) (c) finding

Figure 5: **Some kinds of behaviors during learning process.**

$\gamma = 0.999$, and the robot shifted its body to the better position for getting a shoot. While, in (b), $\gamma = 0.6$, and the robot tried to get a shoot immediately. (a) shows a series of behaviors: first the robot lost the ball, then tried to find it by rotating itself, and finally it dribbled and got a shoot.

5.2 Real System

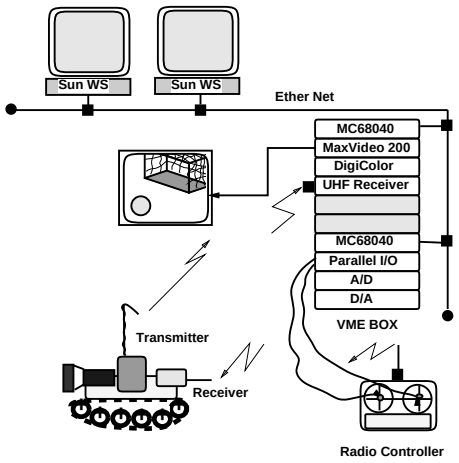
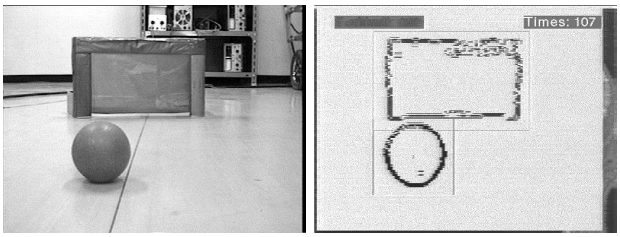


Figure 6: **A configuration of the real system.**

Fig.6 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. In order to simplify and speed up the image processing time, we painted the ball in red and the goal in blue. We constructed the radio control system of the vehicle, following the remote-brain project by Profs. Inaba and Inoue at University of Tokyo [16]. The image processing and the



(a) input image (b) detected image

Figure 7: **Detection of the ball and the goal.**

vehicle control system are operated by VxWorks OS on MC68040 CPU which are connected with host Sun workstations via Ether net. We have shown a picture of the real robot with a TV camera (Sony handy-cam TR-3) and video transmitter in **Fig.3**.

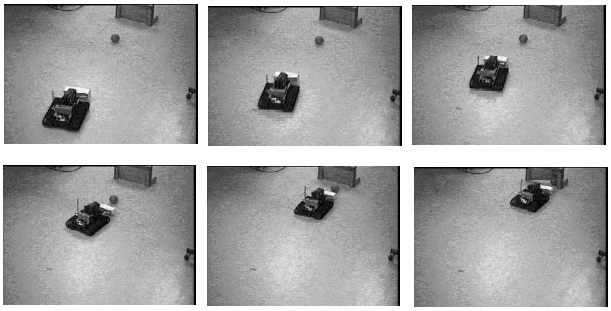


Figure 8: **The robot succeeded in shooting a ball into the goal.**

Fig.7 shows a result of the image processing where the ball and the goal are detected and their positions are calculated in real time (1/30 of a second). **Fig.8** shows a sequence of the shooting images in which the robot succeeded in shooting a ball into the goal.

Table 1 shows the result of state discrimination for the scene shown in **Fig.8**, where the time step (1/30 of a second), state step the robot discriminated, ball state (Left, Right, Center, Disappeared, and Near, Medium, or Far), goal state (in addition to the same states as a ball, Front Oriented, Left Oriented, or Right Oriented), control commands to right and left motors (Forward, Stop, or Backward), and the number of failures of state discrimination. The misunderstood states are with *s. Although error ratio of the state discrimination was very high (about 15%)

Table 1: State-Action data

time step	state step	state		action		errer
		ball	goal	L	R	
1	1	(C,F)	(C,F,Fo)	F	F	
2	2	(R*,F)	(C,F,Fo)	F	F	1
3	3	(D*,D*)	(C,F,Ro*)	B	B	3
4	4	(C,F)	(C,F,Lo*)	B	S	1
5	5	(C,F)	(C,F,Fo)	F	F	
6		(C,F)	(C,F,Fo)	F	F	
7		(C,F)	(C,F,Fo)	F	F	
8		(C,F)	(C,F,Fo)	F	F	
9	6	(C,F)	(C,F,Ro*)	B	S	1
10	7	(C,F)	(C,F,Fo)	F	F	
11	8	(C,F)	(R,M,Fo)	F	F	
12	9	(R,F)	(R,M,Fo)	F	F	
13	10	(R,M*)	(R,F*,Lo*)	F	B	3
14	11	(L*,F)	(R,M,Ro*)	F	S	2
15	12	(L*,F)	(R,M,Fo)	F	S	1
16	13	(R,M)	(R,M,Fo)	S	B	
17	14	(C,M)	(C,M,Fo)	F	F	
18	15	(L,M)	(L,M,Fo)	S	F	
19	16	(L,N)	(L,M,Fo)	B	S	
20		(L,N)	(L,M,Fo)	B	S	
21	17	(L,M*)	(L,M,Fo)	S	F	1
22	18	(L,N)	(L,M,Fo)	B	S	
23		(L,N)	(L,M,Fo)	B	S	
24	19	(C,N)	(C,M,Fo)	F	B	
25	20	(C,M)	(C,M,Fo)	F	F	
26		(C,M)	(C,M,Fo)	F	F	
27	21	(C,M)	(C,N,Fo)	F	S	
28	22	(C,M)	(C,M*,Lo*)	F	S	2
29	23	(C,M)	(C,M*,Ro*)	S	B	2
30	24	(C,F)	(D,D,D)	F	S	

due to noise of image processing, the robot succeeded in shooting a ball into the goal as long as the errors do not occur continuously because the robot has the optimal action value function against the all states.

6 Discussion and Future Works

As a first step towards an autonomous agent capable to generate a purposive behavior, we have studied the feasibility of realizing a shooting behavior with vision. Although we need very longer learning time, the robot has learned to generate a shooting behavior consisting of a series of actions including finding, dribbling, and shooting a ball.

In the followings, we argue the role of vision in the context of the vision-based reinforcement learning.

- During the learning process, the visual information has an important role of state discrimination and eventually action evaluation. Almost of the previous works in the reinforcement learning applications assume the perfect sensors that are often too idealized to apply the real situations. The vision is the most suitable sensor because of its global scope to the environment. Only the problem is how to extract the necessary information to accomplish the task.

- The action value function obtained after the reinforcement learning includes the necessary information for the robot to take the optimal action against each individual state of the environment, and therefore it could be called the environment map.
- Two environments which are different in their appearances could be found similar to each other if we can find the similarity in the action value functions for these environments. A desk or a chair in the office scene and a large rock in the outdoor scene need not be discriminated for a obstacle finder and avoider.
- How to construct a state space is one of the issues in the reinforcement learning scheme. This is called *structural credit assignment problem*. Many existing works in the reinforcement learning construct the state space in such a way that their simulations work well. This sometimes causes unnatural segmentation of the sensory information and mapping to the state space. At least, this can be avoided if the robot uses the visual information because the spatial resolution and the dynamic range of the observed intensities are limited to some extents and therefore the robot cannot discriminate the state beyond these physical constraints.
- In the vision based reinforcement learning scheme, there are two issues related to each other: coarse segmentation of the state space and real-time processing (state mapping and control). In our work, the red ball and the blue goal are easily extracted and their sizes and positions are very coarsely mapped to the state space, that is, "right," "center," or "left" and so on in order to reduce the size of the state space. This contributes to absorb small amount of errors in measuring the positions and the sizes of the ball or the gall in the image captured by the robot. In order to cover this coarseness of the state space, control of the robot action must be done in real-time (1/30 of a second). Since it has the action value function obtained after the learning, the robot can take the optimal action against any situation of the environment. Therefore, even if mis-mapping of the state due to the image noise or the failure of action execution due to the slip between the floor and crawler of the robot happens, the robot succeed in shooting a ball into the goal as long as the frequencies of these mistakes is low (See Table 1).

Although we have other problems such as the temporal credit assignment problem when to give a reward to speed up the learning rate and the scaling problem to apply the learned policy to similar tasks but different environments, we realized that the vision-based reinforcement learning method seems promising in realizing autonomous agent in real world. We are planning to extend the method to multiple players coordination and competition.

References

[1] R. A. Brooks. "A robust layered control system for a mobile robot". *IEEE J. Robotics and Automation*, RA-2:14–23, 1986.

[2] R. A. Brooks. "Elephants don't play chess". In P. Maes, editor, *Designing Autonomous Agents*, pages 3–15. MIT/Elsevier, 1991.

[3] M. J. Mataric. "Integration of representation into goal-driven behavior based robots". *IEEE J. Robotics and Automation*, RA-8:–, 1992.

[4] P. Maes. "The dynamics of action selection". In *Proc. of IJCAI-89*, pages 991–997, 1989.

[5] R. C. Arkin. "Integrating behavioral, perceptual, and world knowledge in reactive navigation". In P. Maes, editor, *Designing Autonomous Agents*, pages 105–122. MIT/Elsevier, 1991.

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

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

[8] P. Maes and R. A. Brooks. "Learning to coordinate behaviors". In *Proc. of AAAI-90*, pages 796–802, 1990.

[9] 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.

[10] S. D. Whitehead and D. H. Ballard. "Active perception and reinforcement learning". In *Proc. of Workshop on Machine Learning-1990*, pages 179–188, 1990.

[11] C. J. C. H. Watkins and P. Dayan. "Technical note: Q-learning". *Machine Learning*, 8:279–292, 1992.

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

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

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

[15] R. A. Brooks and M. J. Mataric. "Real robot, real learning problems". In J. H. Connel and S. Mahadevan, editors, *Robot Learning*, chapter 8. Kluwer Academic Publishers, 1993.

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