Environmental Complexity Control for Vision-Based Learning Mobile Robot

Eiji Uchibe, Minoru Asada and Koh Hosoda
Dept. of Adaptive Machine Systems, Graduate School of Eng.,
Osaka University, Suita, Osaka 565-0871, Japan
uchibe@er.ams.eng.osaka-u.ac.jp

Abstract

This paper discusses how a robot can develop its state vector according to the complexity of the interactions with its environment. A method for controlling the complexity is proposed for a vision-based mobile robot of which task is to shoot a ball into a goal avoiding collisions with a goal keeper. First, we provide the most difficult situation (the maximum speed of the goal keeper with chasing-a-ball behavior), and the robot estimates the full set of state vectors with the order of the major vector components by a method of system identification. The environmental complexity is defined in terms of the speed of the goal keeper while the complexity of the state vector is the number of the dimensions of the state vector. According to the increase of the speed of the goal keeper, the dimension of the state vector is increased by taking a trade-off between the size of the state space (the dimension) and the learning time. Simulations are shown, and other issues for the complexity control are discussed.

1 Introduction

One of the ultimate goals of Robotics and AI is to realize autonomous agents that organize their own internal structure towards achieving their goals through interactions with dynamically changing environments. From a viewpoint of designing robots, there are two main issues to be considered:

- the design of the agent architecture by which a robot develop from the interaction with its environment to obtain the desired behaviors, and
- the policy how to provide the agent with tasks, situations, and environments so as to develop the robot.

The former has revealed the importance of "having bodies" and eventually also a view of the internal observer [7]. In [2], the first issue is focused and a discussion how the robot can develop from the interaction with its environment according to the increase of the complexity of its environment is given in the context of a vision based mobile robot of which task is to shoot a ball into a goal with/without a goal keeper. In this paper, we put more emphasis on the second issue, that is, how to control the environmental complexity so that the robot can efficiently improve its behaviors.

"Shaping by successive approximation" is a well-known technique in psychology of animal behavior [6]. A simple and straightforward analogy to this situation is to design a reward function to accelerate the reinforcement learning. However, this often requires *a priori* precise knowledge about the details of the relationship between the given task and the environment. Instead of providing such knowledge, an alternative called "Leaning from Easy Missions" (LEM) paradigm was proposed [3].

The basic idea of LEM can be extended to more complicated tasks, but more fundamental issues to be considered are how to define complexity of the task and the environment, and how to increase the complexity to develop robots. Since these issues are too difficult to deal with as general ones, a case study on a vision-based mobile robot is given in this paper where the environmental complexity is defined in the context of RoboCup Initiative [4] and a method to control the environmental complexity is proposed. First, we provide the most difficult situation, that is, the maximum speed of the goal keeper with chasing-a-ball behavior, and the robot estimates the full set of state vectors with the order of the vector components according to the contributions to reducing the estimation errors by a method of system identification. The environmental complexity is defined in terms of the speed of the goal keeper while the complexity of the state vector to cope with the environmental complexity is the number of

the dimensions of the state vector. According to the increase of the speed of the goal keeper, the dimension of the state vector is increased by taking a trade-off between the size of the state space (the dimension) and the learning time.

The rest of the paper is organized as follows: first we give an overview of the whole learning system, and basics of the reinforcement learning, especially Q-learning, Next, a method for efficient learning and development coping with the increase of the task environment complexity is proposed. Then, an example task of shooting with avoiding a goal keeper is introduced. The proposed method is applied to scheduling the speed of the goal keeper for the efficient development of the learner that attempting at coping with new situations by adding a new axis in its state space. Finally, the preliminary experiments are shown, and other issues for the complexity control are discussed.

2 An Overview of The Whole System

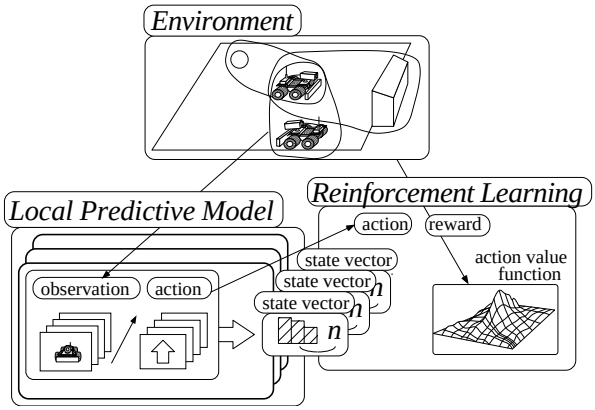


Figure 1: An overview of the whole system

Figure 1 shows an overview of the whole system consisting of a local predictive model and a learning architecture. The local predictive model outputs the state vector list in the order of the value of the estimated correlation coefficient with estimation errors. These state vectors are used to construct the state space for the reinforcement learning method to be applied in multi agent environment. About the details of the whole system, one can find other publications [9, 10]. Here, we focus on how to accelerate the Q-learning by appropriately increasing the environmental complexity. The rest of this section briefly explains the basics of state vector estimation and the reinforcement learning.

2.1 State Vector Estimation

In order to accelerate the learning according to the increase of the environmental complexity, it needs a mechanism to measure the complexity based on its experience. As such a mechanism, a local predictive model [9] is considered which estimates the relations between the learner's behaviors and the other agents through interactions (observation and action). In order to construct the local predictive model of other agents, Akaike's Information Criterion(AIC) [1] is applied to the result of Canonical Variate Analysis(CVA) [5]. We just briefly explained the method (for the details of the local predictive model, see [9, 10]).

CVA uses a discrete time, linear, state space model as follows:

$$\begin{aligned} \boldsymbol{x}(t+1) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t), \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t), \end{aligned} \tag{1}$$

where $\boldsymbol{x}(t)$, $\boldsymbol{u}(t) \in \Re^m$ and $\boldsymbol{y}(t) \in \Re^q$ denote state vector, action code vector, and observation vector respectively. $\boldsymbol{A} \in \Re^{n\times n}$, $\boldsymbol{B} \in \Re^{n\times m}$, $\boldsymbol{C} \in \Re^{q\times n}$, and $\boldsymbol{D} \in \Re^{q\times m}$ represent matrices. CVA estimates a state vector $\boldsymbol{x}$ which is a linear combination of the previous observation and action sequences as follows:

$$\boldsymbol{x}(t) = [\boldsymbol{I}_n \ \boldsymbol{0}]\boldsymbol{U}\boldsymbol{p}(t), \tag{2}$$

where

$$\boldsymbol{p}(t) = [\boldsymbol{u}(t-1) \cdots \boldsymbol{u}(t-l)\, \boldsymbol{y}(t-1) \cdots \boldsymbol{y}(t-l)]^T,$$

and $\boldsymbol{U} \in \Re^{l(m+q)\times l(m+q)}$ is a matrix which is calculated by CVA.

2.2 Basics of Reinforcement Learning

After estimating the state space model given by Eq. (1), the agent begins to learn behaviors using a reinforcement learning method. Q learning [11] is a form of reinforcement learning based on stochastic dynamic programming. It provides robots with the capability of learning to act optimally in a Markovian environment.

In the previous section, appropriate dimension n of the state vector $\boldsymbol{x}(t)$ is determined, and the successive state is predicted. Therefore, we can regard an environment as Markovian. A simple version of Q learning algorithm is shown as follows:

1. Initialize $Q(x, u)$ to 0s for all combination of $\boldsymbol{X}$ and $\boldsymbol{U}$.

2. Perceive current state x.

3. Choose an action u according to the action value function.

4. Execute an action u in the environment. Let the next state be x' and immediate reward be r.

5. Update the action value function from x, u, x', and r,

$$\begin{aligned} Q_{t+1}(x,u) &= (1-\alpha_t)Q_t(x,u) \\ &+ \alpha_t(r+\gamma \max_{u' \in \boldsymbol{U}} Q_t(x',u')) \quad (3) \end{aligned}$$

where α_t is a learning rate and γ is a fixed discounting factor between 0 and 1.

6. Return to 2.

3 The Method for Efficient Learning and Development

One can use all the state vectors to make the robot learn, but it would take enormously long time due to the large size of the state space. Instead of using the all vectors, one can start with a small size of the state vector set first and increase the dimension of the state space in the following stages. The action value function in the previous stage works as a priori knowledge so as to accelerate the learning. In order to transfer the knowledge smoothly, the state spaces in both the previous and current stages should be consistent with each other. Therefore, the robot should have a full list of the state vectors available in advance, and selects one among them at the periods when the robot no longer can cope with the changing environment with the current state vector set.

An algorithm to control the increase of the environmental complexity is given as follows:

1. Collect many sequences of data during action executions in the most complex task environment.

2. Construct the local predictive model to the data and output the state vector lists with estimation errors.

3. Set up the performance criterion.

4. Start with the minimum state vector set, say one or two dimensions for the lowest complexity of the task environment.

5. Keep the complexity until the robot learns the desired behavior (reach the performance criterion).

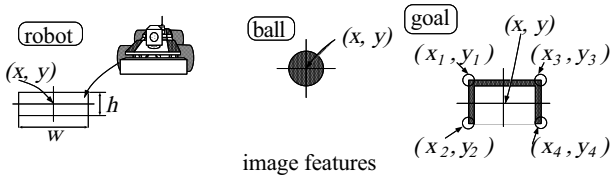
6. If the robot reaches the performance criterion, increase the complexity and return Step 5. Else, increase the dimension of the state space (add a new axis) and return Step 5.

As a learning method, we use modular reinforcement learning [8] based on Q-learning with the state space specified. The modular reinforcement learning can coordinate multiple behaviors (in the following, shooting behavior and avoiding one) taking account of a trade-off between the learning time and the performance.

4 Experimental Results

4.1 Task and Assumptions

We apply the proposed method to a simplified soccer game including two agents [8]. One is a learner to shoot a ball into a goal, and the other is a goal keeper of which speed is a control parameter in the environment complexity. Each agent has a single color TV camera and observes output vectors shown in Figure 2. The dimension of the observed vector about the ball, the goal, and the other robot are 4, 11, and 5 respectively.



robot	ball	goal
area	area	area
center position	center position	center position
height	radius	4 corners
width		

Figure 2: Image features of the ball, goal, and agent

Two robots move around using a 4-wheel steering system. The effects of an action against the environment can be informed to the agent only through the visual information except the reward that is given by the environment (top down signal). Figure 3 shows a scene of two real robots and the environment. As motor commands, each agent has 7 actions such as go straight, turn right, turn left, stop, and go backward. Then, the input $\boldsymbol{u}$ is defined as the 2 dimensional vector as

$$\boldsymbol{u}^T = [v \;\; \phi], \quad v, \phi \in \{-1, 0, 1\},$$

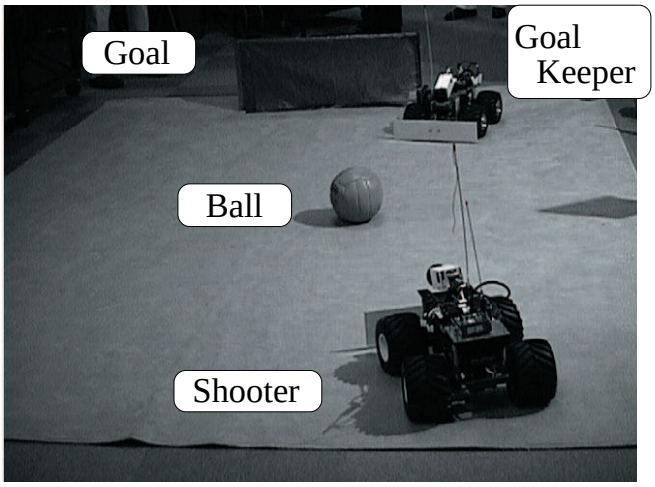


Figure 3: Two real robots and the environment

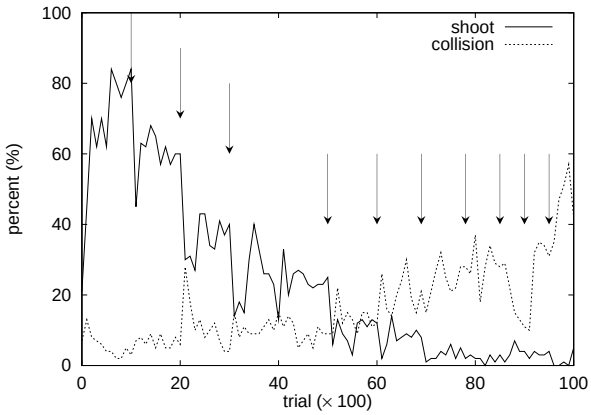


Figure 4: The dimension of the state vector n is one (the minimum dimension).

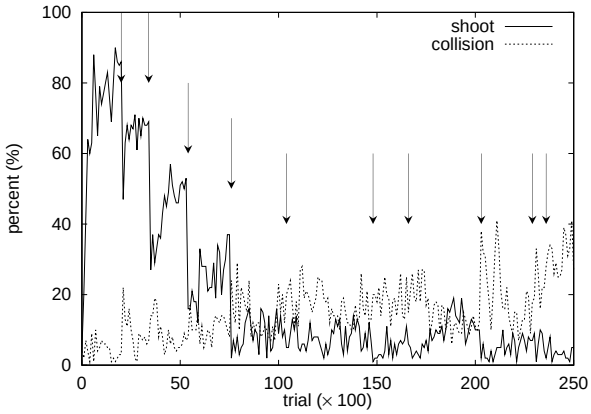


Figure 5: The dimension of the state vector n is 2.

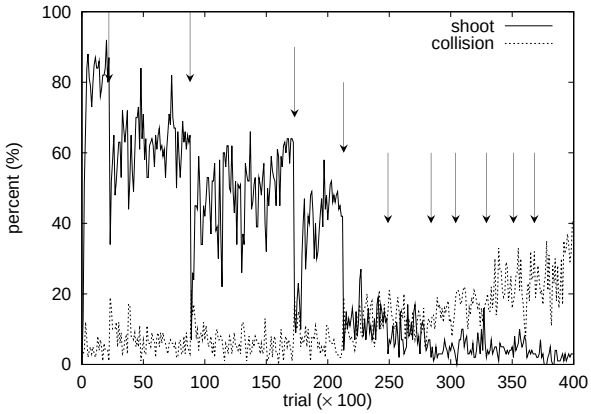


Figure 6: The dimension of the state vector n is 3.

where v and ϕ are the velocity of motor and the angle of steering respectively and both of which are quantized.

We assume that the goal keeper has a basic behavior of moving to the ball, but its speed can be controlled as the complexity parameter.

4.2 Settings

We assign a reward value 1 when the ball was kicked into the goal or 0 otherwise. On the other hand, a reward value -0.3 is given to the robot when two robots make a collision between them. Discounting factor γ is 0.9.

To speed up the learning time, we select actions using the probability based on *semi uniform undirected exploration*. In this method, the learning agent executes random actions with a fixed probability. We set the probability of selecting a random action at 10 %.

4.3 Speed Control for the Goal Keeper with Fixed Dimension

At first, we demonstrate the experiments to control the complexity of the interactions in case of the fixed dimension of the estimated state vector about the goal keeper. The learning robot collects sequences of observation and action with the highest complexity, that is, the maximum speed $v_{\max}$ of the goal keeper, and applied the local predictive model to the obtained data. As a result, we obtained the list of the state vector for the goal keeper and others. The dimension of the estimated state vector of the goal keeper, the ball and the goal is 4, 4 and 2, respectively. The learning

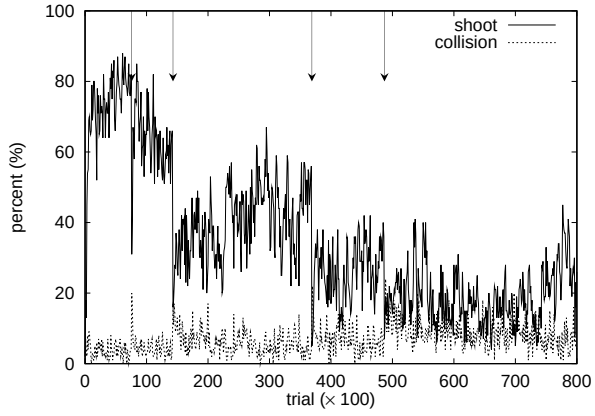


Figure 7: The dimension of the state vector n is 4 (the maximum dimension).

robot chooses the dimension of the only state vector about the goal keeper (other vectors are remained unchanged) which is estimated by the local predictive model to cope with the change of the complexity of the interaction.

Figures $4 \sim 7$ show graphs of the performance data (success rates of shooting and collision avoidance) in terms of the speed of the goal keeper with fixed dimension of the state vector for the goal keeper (from 1 to 4). The speed is increased when the robot achieves the pre-specified success rate (80%) or no improvement can be seen. The arrows show the time when the speed of the goal keeper is changed (10 % speed increase of the maximum motion speed $v_{\max}$ from 0 (stationary)). In spite of the number of dimensions, the best success rate of shooting is about 80 %. However it takes much time for learning agent to acquire the best performance when the dimension of the state space for the goal keeper increases. In Figure 7, the performance data until the speed of $0.4v_{\max}$ is shown because of the space limit. As we can see from the Figures $4 \sim 7$,

- the success rate of shooting becomes worse when $v/v_{\max} > 0.2$, and
- the collision rate is larger than success one of shooting when $v/v_{\max} > 0.4$.

The learning agent has to take account of the trade off between shooting behavior and avoiding behavior while the goal keeper only pushes the ball. Therefore, the learning agent might not accomplish the shooting task if the goal keeper moves quickly.

4.4 Speed Control for the Goal Keeper with Variable Dimensions

Figure 8 shows the result of the speed control for the efficient learning. Short and long arrows indicate the times to increase the speed of the goal keeper and the dimension of the state vector, respectively. We set up 50% performance criterion by which the timing of the speed increase of the goal keeper is decided. Compared with Figures $4 \sim 7$, we may conclude that the fewer dimensions of the state space contribute to the reduction of the learning time but less performance and vice versa. For example, one dimensional state vector cannot cope with $0.2v_{\max}$ while two dimensional state vector can not represent the situation with $0.3v_{\max}$ for the learner to learn shooting behaviors. If we start with one dimension case and step up the dimension, we also give up $0.4v_{\max}$ but with four dimensions the collision rate is much less than the success rate around 15,000 trials (See Figure 7).

Our proposed scheduling method can achieve the almost the same performance faster than the case of learning by the maximum dimension of the state vector from the beginning. We suppose that the reasons why our method can achieve the task faster are as follows. First, the time needed to acquire an optimal behaviors mainly depends on the size of the state space, which are determined by the dimension of the state vector estimated by the local predictive model. Our method assigns the appropriate dimension of the state vector according to the complexity while the full dimension of the state space (Figure 7) is redundant in the early state of learning. Second, since our proposed method utilizes the action value function which is previously acquired as the initial value, it can reduce the learning time. In other words, our method consider not only the size of the state space according to the complexity but also the initial values of the action value function which is usually initialized zeros. Finally, we show the example of an acquired behavior in Figure 9. The two lines emerged from the agent show its visual angle.

5 Discussion

We have shown the method of controlling the environmental complexity along with a simplified soccer task. There are two main issues to be considered. First, the number of control parameters is one in our experiments, but generally multiple, each of which is related to each other. Even in the example task, the

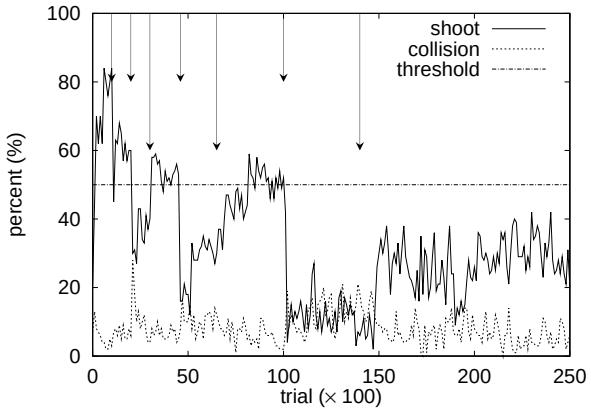


Figure 8: Result of the proposed method

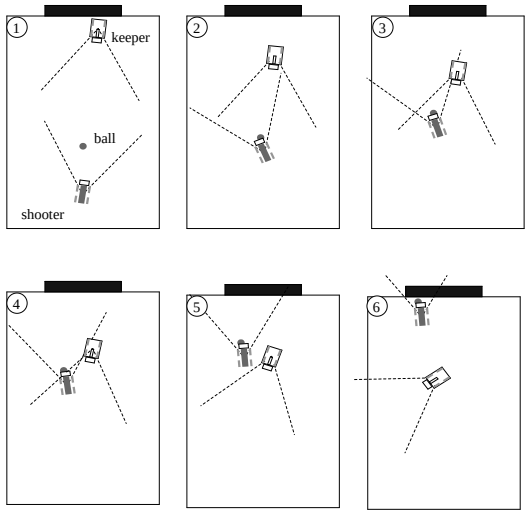


Figure 9: Visualization of the learning agent's policy at the end of a successful trial

speed of the learner, the dimensions of the state space, the resolution of the each dimension (fixed (3 partitions) in the experiments) and the initial configurations of the ball, the goal, the learner, and the goal keeper should be considered together with the speed of the goal keeper. In such a case, since designer cannot completely understand the relationships among them, it seems difficult to decide how to control the complexity completely.

Then, the second issue is revealed. To cope with unknown complexity, the robot should estimate the state vectors anytime when the task performance becomes worse. However, this causes inconsistency in state vector sets between the current and next learning stages. Therefore, the knowledge transfer is limited to the initial controller (action selection) and the robot needs much more memory and the learning time. Since this is against resource bounded condition, we should develop a new method which can take account of this trade-off.

References

[1] H. Akaike. A new look on the statistical model identification. *IEEE Trans. AC-19*, pp. 716–723, 1974.

[2] M. Asada. An agent and an environment: A view of "having bodies" – a case study on behavior learning for vision-based mobile robot –. In *Proc. of 1996 IROS Workshop on Towards Real Autonomy*, pp. 19–24, 1996.

[3] M. Asada, S. Noda, S. Tawaratumida, and K. Hosoda. Purposive behavior acquisition for a real robot by vision-based reinforcement learning. *Machine Learning*, 23:279–303, 1996.

[4] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara. Robocup a challenge problem for ai. *AI Magazine*, 18(1):73–85, 1997.

[5] W. E. Larimore. Canonical variate analysis in identification, filtering, and adaptive control. In *Proc. 29th IEEE Conference on Decision and Control*, pp. 596–604, Honolulu, Hawaii, December 1990.

[6] B. Schwartz. *Psychology of Learning and Behavior: Third Edition*. W. W. Norton, NY, London, 1989.

[7] J. Tani. Cognition of robots from dynamical systems perspective. In *Proc. of 1996 Workshop on Towards Real Autonomy*, pp. 51–59, 1996.

[8] E. Uchibe, M. Asada, and K. Hosoda. Behavior coordination for a mobile robot using modular reinforcement learning. In *Proc. of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1329–1336, 1996.

[9] E. Uchibe, M. Asada, and K. Hosoda. State space construction for behavior acquisition in multi agent environments with vision and action. In *Proc. of International Conference on Computer Vision*, pp. 870–875, 1998.

[10] E. Uchibe, M. Asada, and K. Hosoda. Cooperative behavior acquisition in multi mobile robots environment by reinforcement learning based on state vector estimation. In *Proc. of IEEE International Conference on Robotics and Automation*, 1998.

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