Vision-based Behavior Learning and Development for Emergence of Robot Intelligence

M. Asada, K. Hosoda, and S. Suzuki
Dept. of Adaptive Machine Systems, Graduate School of Engineering
Osaka University, Suita, Osaka 565 (Japan)
email: asada@ams.eng.osaka-u.ac.jp

Abstract

This paper focuses on two issues on learning and development; a problem of state-action space construction, and a scaling-up problem. The former is mainly related to sensory-motor mapping and its abstraction, and we show two our methods for the state and action space construction for reinforcement learning. For the latter issue, we attempt to define the environmental complexity based on the relationships between observations and self motions. Based on this view, we introduce a method which can cope with the complexity of multi-agent environment by a combination of a state vector estimation process and a reinforcement learning process based on the estimated vectors. As example tasks in our work, we adopt the domain of soccer robots, RoboCup [1]. Computer simulations and real robot experiments are given.

1 Introduction

The ultimate goal of our research is to design the fundamental internal structure inside physical entities having their bodies (robots) which can emerge complex behaviors through the interactions with their environments. In order to emerge the intelligent behaviors, physical bodies have an important role of bringing the system into *meaningful* interaction with the physical environment – complex, uncertain, but with automatically consistent set of natural constraints. This facilitates the correct agent design, learning from the environment, and rich meaningful agent interaction. The meanings of "having a physical body" can be summarized as follows:

1. Sensing and acting capabilities are not separable, but tightly coupled.
2. In order to accomplish given tasks, the sensor and actuator spaces should be abstracted under the resource bounded conditions (memory, processing power, controller etc.).
3. The abstraction depends on both the fundamental embodiments inside the agents and the experiences (interactions with their environments).
4. The consequences of the abstraction are the agent-based subjective representation of the environment, and its evaluation can be done by the consequences of behaviors.
5. In real world, both inter-agent and agent-environment interactions are asynchronous, parallel and arbitrarily complex. The agent should cope with increasing complexity of the environment to accomplish the given task at hand.

In this paper, we focus on two issues on learning and development; a problem of state-action space construction, and a scaling-up problem. The former is mainly related to 2 and 3, and we show two our methods for the state and action space construction for reinforcement learning. One is based on an off-line learning method [2] and the other on-line one [3].

The latter issue is closely related to 4 and 5, and we attempt to define the environmental complexity based on the relationships between observations and self motions. Based on this view, we introduce a method which can cope with the complexity of multi-agent environment by a combination of a state vector estimation process and a reinforcement learning process based on the estimated vectors [4].

As example tasks in our work, we adopt the domain of soccer robots, RoboCup, which is an attempt to foster intelligent robotics research by

providing a standard problem where a wide range of technologies can be integrated and examined [1].

The remainder of this article is structured as follows. We first give an explanation of the problem of state-action space construction along with our real robot experiments in the context of reinforcement learning. Then, we show our method to cope with more complicated tasks in multi-agent environment. Finally, we give a conclusion.

2 A Problem of State-Action Space Construction

Reinforcement learning [5, 6] has 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. In such robot learning methods, a robot and an environment are generally 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.

To apply robot learning methods such as reinforcement learning to real robot tasks, we need a well-defined state-action space by which the robot learns to select an adequate action for the current state to accomplish the task at hand. Traditional notions of state and action in the existing applications of the reinforcement learning schemes fit nicely into deterministic state transition models (e.g. one action is forward, backward, left, or right, and the states are encoded by the locations of the agent). However, it seems difficult to apply such deterministic state transition models to real robot tasks. In real world, everything changes asynchronously [7]. Therefore, the construction of state-action space is one of the most important issues in robot learning.

Generally, the design of the state-action space in which the necessary and sufficient information to accomplish a given task should be included depends on the capabilities of agent sensing and acting. The abstraction process from sensory information to a state seems to depend on the process from motor commands to an action, and *vice versa*. This resembles the well-known "chicken and egg problem" that is difficult to be solved (see Figure 1). Therefore, we need a sort of constraint to solve the problem.

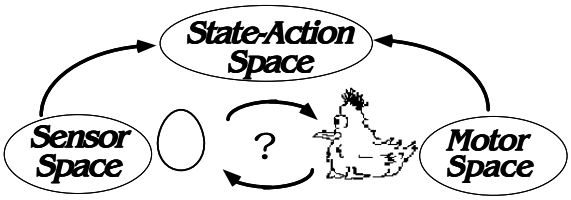


Figure 1: The inter-dependence between sensor and motor spaces from a viewpoint of state-action space construction

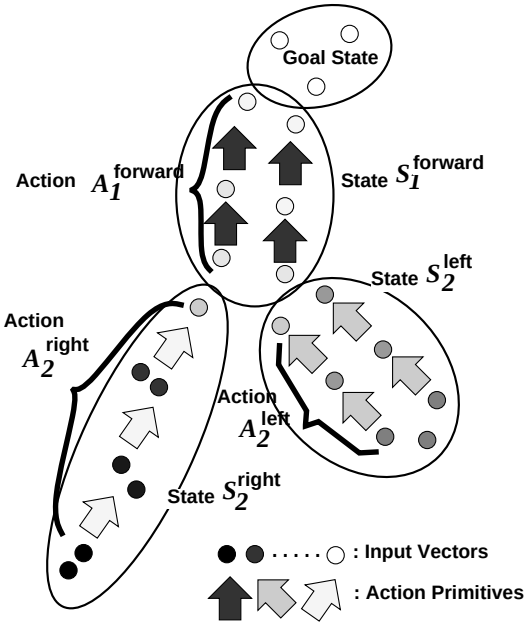


Figure 2: A basic idea of state-action space construction

2.1 An Off-Line Learning Method

Basic ideas of our first approach to cope with this problem are:

- we define an action primitive as a motor command to be executed during a fixed time interval, and an input vector as sensory data of the consequence of the action primitive, and

- we define a state as a cluster of input vectors from which the robot can reach the goal state (or the state already obtained) by a sequence of one kind action primitive regardless of its length, and one action as this sequence of action primitives.

Figure 2 shows the basic idea of the state-action space construction. The initial state space consisting of the goal state and the other is iteratively separated into several states.

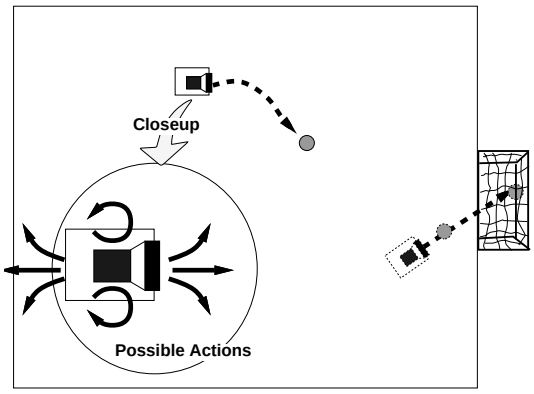


Figure 3: Task and environment

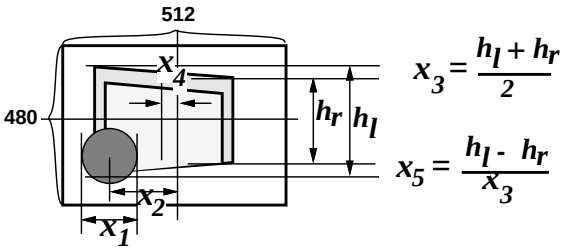


Figure 4: Input vector consisting of five parameters

The method is applied to a soccer robot which tries to shoot a ball into a goal (see Fig.3). The size of the observed image is 512 by 480 pixels, and the center of image is the origin of the image coordinate system (see Figure 4). An input vector $\boldsymbol{x}$ for a shooting task consists of:

- x_1: the size of the ball, the diameter that ranges from 0 to about 270 pixels,
- x_2: the position of the ball ranging from -270 to +270, considering the partial observation,

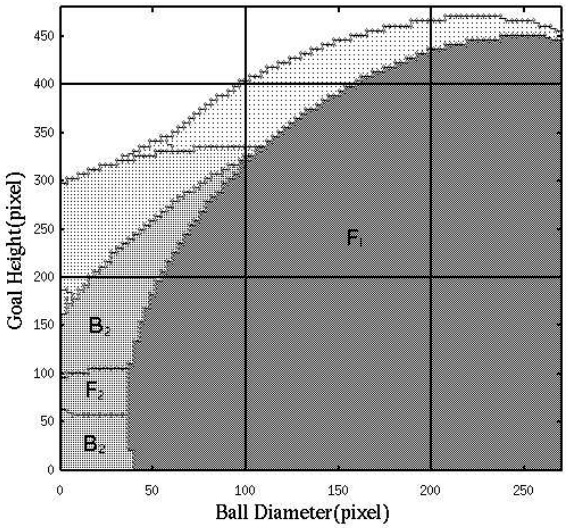


Figure 5: 2-D projection of the result of state space construction

- x_3: the size of the goal, the height average of the left and right poles (or ends in image) that ranges from 0 to 480 pixels,
- x_4: the position of the goal, the average of the positions of the left and right poles (or ends in image) that ranges from -256 to +256, and
- x_5: the orientation of the goal, the ratio of the height difference between the left and right poles (or ends) to the size of the goal x_3. x_5 ranges from -1.00 to +1.00.

Figure 5 shows the result in which the final state space is projected into two dimensional space in terms of the ball size and the goal size (when their positions are frontal and the orientation of the goal is horizontal). The intensity indicates the order of the division: the darker is the earlier. Labels "F" and "B" indicate the motions of forward and backward, respectively, and subscript shows the number of state transitions towards the goal. Grid lines indicate the boundaries divided by programmer in the previous work [8]. The remainder of the state space in Figure 5 corresponds to infeasible situations such as "the goal and the ball are observed at the center of image, and the size of the goal is large, but that of the ball is small" although we had not recognized such a meaningless state in the previous work. As we can see, the sensor space categorization by the

proposed method is quite different from the one designed by the programmer (rectangular grids) in the previous work [8].

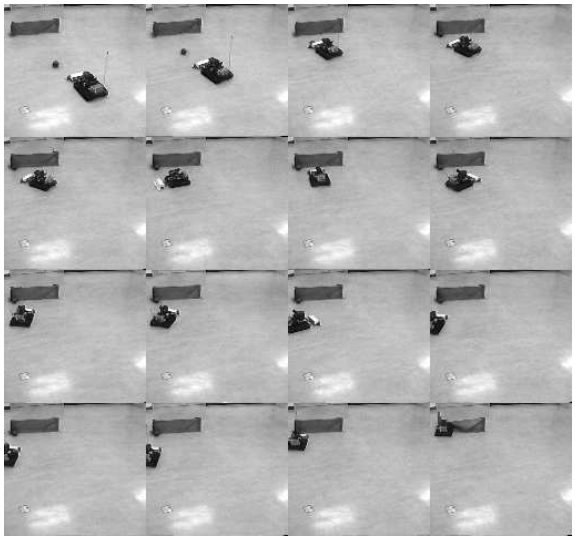


Figure 6: The robot succeeded in finding and shooting a ball into the goal

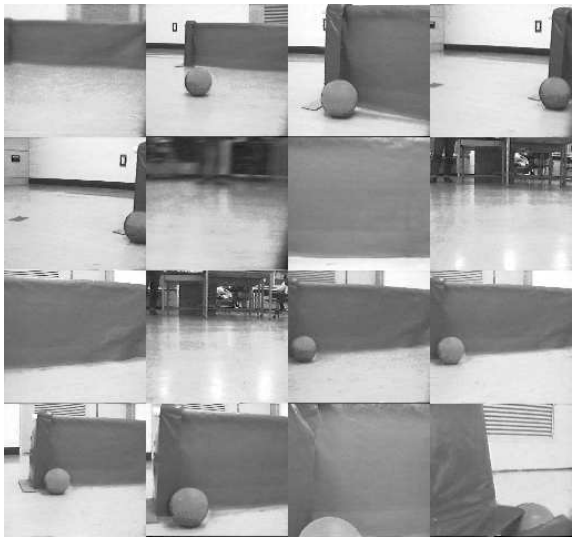


Figure 7: Images taken by the robot during the task execution

We applied the method to a real robot environment. The success ratio is worse than the simulation because of the disturbances due to several causes such as eccentricity of the ball centroid and slip of the tires that make the ball or the robot move into unpredictable directions. Figure 6 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 shows a sequence of images taken by the robot during the task execution shown in Figure 6. Note that the backward motion for retry is just the result of learning and not hand-coded.

2.2 An On-line Learning Method

The above method needs sufficient amount of uniformly sampled data to construct the state space suitable for the robot to perform the given task, and therefore, does not cope with dynamic changes happened in the environment. These problems are resolved by the second approach [3] which obtains a purposive behavior within less learning time by incrementally segmenting the sensor space based on the experiences of the robot. The incremental segmentation is performed by constructing local models in the state space, which is based on the function approximation of the sensor outputs to reduce the learning time, and the reinforcement signal to emerge a purposive behavior. They applied their method to the same task as in [2]. The basic ideas are as follows:

1. Set up a state space consists of two states; the goal state and the other.

2. Apply
   function approximation to the changes of the input vectors caused by action primitives. If the function approximation cannot cope with these changes, then segment the states into two and apply the function approximation to a new state. This process might cause to merge a state with one of the separated states. These processes can reduce ineffective explorations.

3. Initialize the action-value for the new state, and apply the reinforcement learning. The learning time is very short because the number of states to be updated is small.

4. Apply stochastic action selection to cope with dynamic change of the environment.

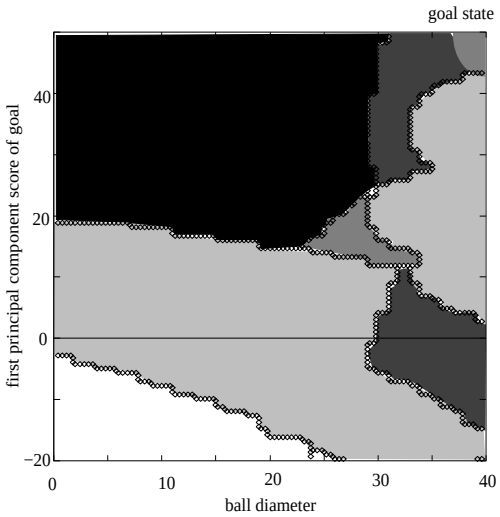


Figure 8: Obtained state space

Figures 8, 9, and 10 show the experimental results. Figure 8 shows a projection of the state space after 1,110 trials, where the state space in term of ball size and goal size is indicated when the position of the ball and the goal are center of the screen and the orientation of the goal is frontal. As we can see the shape of the resultant state space is complicated and quite different from the previous result (see Figure 5). Figure 9 indicates the changes of the success rate and the number of states in the case that the ball size is suddenly changed twice at the 500th trial. These suggest that the method cope with nonlinear mapping between states and actions and

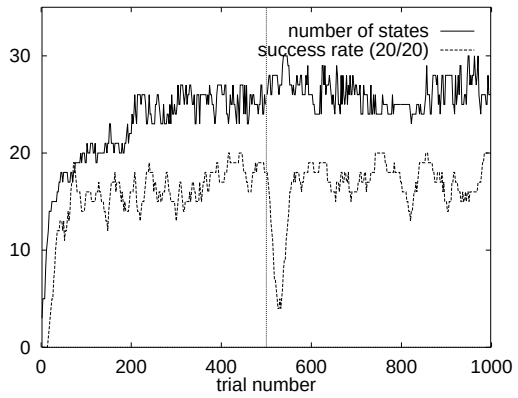


Figure 9: Success rate and the number of states

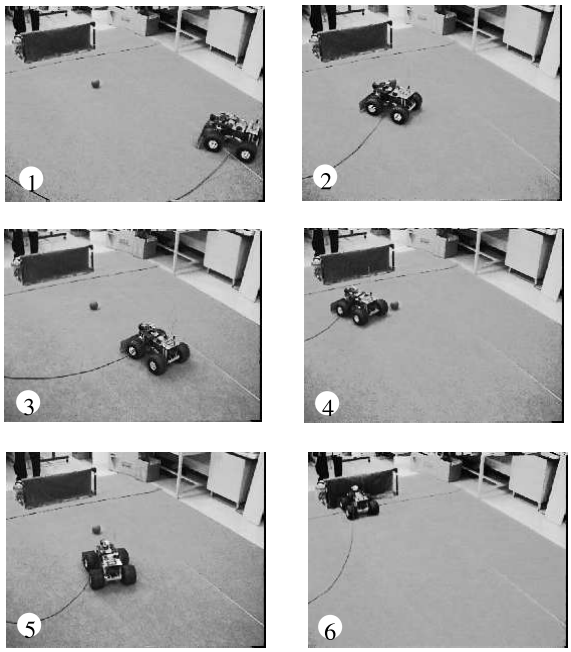


Figure 10: The robot succeeded in shooting a ball into the goal

deal with dynamic change of the environment.

Figure 10 shows how the robot tries to shoot a ball into the goal. Because of the sensor noise and the uncertainty of the motor commands, the robot often misunderstands the states, and takes wrong actions, therefore it fails to do the task. ① indicates that the robot is going to shoot a ball into the goal and moves forward. But it fails to kick the ball at ② because the speed is too high to turn. Eventually, the ball is occluded by the robot in ②. Then, it goes back left so that it can shoot a ball at ③. But it fails again at ④. Then it goes back left again at ⑤. After all, the robot shoots the ball into the goal successfully at ⑥.

3 A Scaling-Up Problem

Since each species of animals can be regarded to have its own intelligence, difference of intelligence seems to depend on the agent (capabilities in sensing, acting, and cognition) and its environment. If agents have the same bodies, differences or levels in intelligence can occur in the complexity of interactions with their environments. In case of our soccer playing robot with vision, the complexity of interactions may change due to other agents in the field such as common side players,

opponents, judges and so on. In the following, we attempt at showing our view about the levels of complexity of interactions, especially from a viewpoint of the existence of other agents.

1. **Self body and Static Environment:** The self body or static environment can be defined in a sense that the observable parts of which changes in the image plane can be directly correlated with the self motor commands (ex. looking at your hand showing voluntary motion, or observing an optical flow of the environment when changing your gaze). Theoretically, discrimination between "self body" and "static environment" is a hard problem because the definition of "static" is relative and depends on the selection of the reference (the base coordinate system) which also depends on the context of the given task. Usually, we suppose the natural orientation of the gravity, which provides the orientation of the ground coordinate system.

2. **Passive agents:** As a result of actions of the self or other agents, passive agents are movable or can be stopped. A ball is a typical one. As long as they are stationary, they can be categorized into the static environment. But, not so simple correlation with motor commands as the self body or the static environment can be expected when they are in motion.

3. **Active (other) agents:** Active other agents do not have a simple and straightforward relationship with the self motions. In the early stage, they are treated as noise or disturbance because of not having direct visual correlation with the self motor commands. Later, they can be found as having more complicated and higher correlation (coordination, competition, and others). The complexity is drastically increased.

According to the complexity of the environment, the internal structure of the robot should be higher and more complex to emerge various intelligent behaviors. We show one of such structure coping with the complexity of agent-environment interactions with real robot experiments and discuss the future issues.

3.1 A More Complicated Task in Multi-Agent Environment

In a multi-agent environment, the conventional reinforcement learning algorithm does not seem applicable because the learner's sensory information may change regardless of the learner's motion due to the motion of other active agents in the environment. Therefore, the learner cannot predict the other agent behaviors correctly unless explicit communication is available. It is important for the learner to discriminate the strategies of the other agents and to predict their movements in advance to learn the behaviors successfully.

The existing methods in multi agent environments (ex., [9],[10],[11],[12],[13] and so on.) need state vectors in order for the learning to converge. However, it is difficult to obtain a reasonable analytical model in advance. Therefore, the modeling architecture is required to make the reinforcement learning applicable.

Here, we show a method which estimates the relationship between the learner's behaviors and the other agents through interactions (observation and action) using the method of system identification. In order to construct the local predictive model of other agents, we apply Akaike's Information Criterion(AIC) [14] to the result of Canonical Variate Analysis(CVA) [15], which is widely used in the field of system identification. The local predictive model is constructed based on the observation and action of the learner (observer).

We apply the proposed method to a simple soccer-like game in a physical environment. The task of the agent is to shoot a ball which is passed back from the other agent. Since the environment consists of the stationary agents (the goal and the line), a passive agent (the ball) and an active agent (the passer), the learner has to construct the adequate models for these agents. After constructing the models and estimating their parameters, the reinforcement learning is applied in order to acquire purposive behaviors. The proposed method can cope with the moving ball because a state vector for learning is selected appropriately so as to predict the successive steps.

Figure 11 shows an overview of the proposed method consisting of local predictive models and reinforcement learning architecture. At first, the learning agent collects the sequence of sensor outputs and motor commands to construct the local predictive models. By approximating the relationship between inputs (learner's action) and

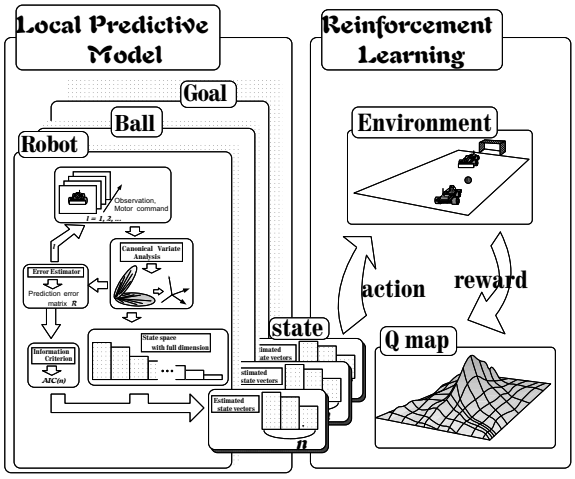


Figure 11: An overview of the proposed method

outputs (observation), the local predictive model gives the learning agent not only the successive states of the agent but also the priority of state vectors, which means that first a few vectors might be sufficient to predict the successive states.

The flow of the proposed method is summarized as follows:

1. Collect the observation vectors and the motor commands.
2. Estimate the state space with the full dimension directly from the observations and motor commands (Section 3.1.1).
3. Determine the dimension of the state vectors which is the result of the trade off between the error and the complexity of the model.
4. Apply the reinforcement learning based on the estimated state vectors.

3.1.1 *Canonical Variate Analysis*

A number of algorithms to identify multi-input multi-output (MIMO) combined deterministic-stochastic systems have been proposed [16]. In contrast to 'classical' algorithms such as PEM (Prediction Error Method), the subspace algorithms do not suffer from the problems caused by a priori parameterizations. Larimore's Canonical Variate Analysis (CVA) [15] is one of such algorithms, which uses canonical correlation analysis to construct a state estimator

Let $\boldsymbol{u}(t) \in \Re^m$ and $\boldsymbol{y}(t) \in \Re^q$ be the input and output generated by the unknown system

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

with

$$E\left\{ \begin{bmatrix} \boldsymbol{w}(t) \\ \boldsymbol{v}(t) \end{bmatrix} \begin{bmatrix} \boldsymbol{w}^T(\tau) & \boldsymbol{v}^T(\tau) \end{bmatrix} \right\} = \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{S} \\ \boldsymbol{S}^T & \boldsymbol{R} \end{bmatrix} \delta_{t\tau},$$

and $\boldsymbol{A}$, $\boldsymbol{Q} \in \Re^{n\times n}$, $\boldsymbol{B} \in \Re^{n\times m}$, $\boldsymbol{C} \in \Re^{q\times n}$, $\boldsymbol{D} \in \Re^{q\times m}$, $\boldsymbol{S} \in \Re^{n\times q}$, $\boldsymbol{R} \in \Re^{q\times q}$. $E\{\cdot\}$ denotes the expected value operator and $\delta_{t\tau}$ the Kronecker delta. $\boldsymbol{v}(t) \in \Re^q$ and $\boldsymbol{w}(t) \in \Re^n$ are unobserved, Gaussian-distributed, zero-mean, white noise vector sequences. CVA uses a new vector $\boldsymbol{\mu}$ which is a linear combination of the previous input-output sequences since it is difficult to determine the dimension of $\boldsymbol{x}$. Eq.(1) is transformed as follows:

$$\begin{bmatrix} \boldsymbol{\mu}(t+1) \\ \boldsymbol{y}(t) \end{bmatrix} = \boldsymbol{\Theta} \begin{bmatrix} \boldsymbol{\mu}(t) \\ \boldsymbol{u}(t) \end{bmatrix} + \begin{bmatrix} \boldsymbol{T}^{-1}\boldsymbol{w}(t) \\ \boldsymbol{v}(t), \end{bmatrix}, \quad (2)$$

where

$$\hat{\boldsymbol{\Theta}} = \begin{bmatrix} \boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T} & \boldsymbol{T}^{-1}\boldsymbol{B} \\ \boldsymbol{C}\boldsymbol{T} & \boldsymbol{D} \end{bmatrix}, \quad (3)$$

and $\boldsymbol{x}(t) = \boldsymbol{T}\boldsymbol{\mu}(t)$. We follow the simple explanation of the CVA method.

1. For $\{\boldsymbol{u}(t), \boldsymbol{y}(t)\}$, $t = 1, \cdots N$, construct new vectors

$$\boldsymbol{p}(t) = \begin{bmatrix} \boldsymbol{u}(t-1) \\ \vdots \\ \boldsymbol{u}(t-l) \\ \boldsymbol{y}(t-1) \\ \vdots \\ \boldsymbol{y}(t-l) \end{bmatrix}, \quad \boldsymbol{f}(t) = \begin{bmatrix} \boldsymbol{y}(t) \\ \boldsymbol{y}(t+1) \\ \vdots \\ \boldsymbol{y}(t+k-1) \end{bmatrix},$$

2. Compute estimated covariance matrices $\hat{\boldsymbol{\Sigma}}_{pp}$, $\hat{\boldsymbol{\Sigma}}_{pf}$ and $\hat{\boldsymbol{\Sigma}}_{ff}$, where $\hat{\boldsymbol{\Sigma}}_{pp}$ and $\hat{\boldsymbol{\Sigma}}_{ff}$ are regular matrices.
3. Compute singular value decomposition

$$\begin{aligned} \hat{\boldsymbol{\Sigma}}_{pp}^{-1/2}\hat{\boldsymbol{\Sigma}}_{pf}\hat{\boldsymbol{\Sigma}}_{ff}^{-1/2} &= \boldsymbol{U}_{aux}\boldsymbol{S}_{aux}\boldsymbol{V}_{aux}^T, \quad (4) \\ \boldsymbol{U}_{aux}\boldsymbol{U}_{aux}^T = \boldsymbol{I}_{l(m+q)},\ \boldsymbol{V}_{aux}\boldsymbol{V}_{aux}^T &= \boldsymbol{I}_{kq}, \end{aligned}$$

and $\boldsymbol{U}$ is defined as:

$$\boldsymbol{U} := \boldsymbol{U}_{aux}^T\hat{\boldsymbol{\Sigma}}_{pp}^{-1/2}.$$

4. The n dimensional new vector $\boldsymbol{\mu}(t)$ is defined as:

$$\boldsymbol{\mu}(t) = [\boldsymbol{I}_n \ \ 0]\boldsymbol{U}\boldsymbol{p}(t), \tag{5}$$

5. Estimate the parameter matrix $\boldsymbol{\Theta}$ applying the least square method to Eq (2).

Strictly speaking, all the agents do in fact interact with each other, therefore the learning agent should construct the local predictive model taking these interactions into account. However, it is intractable to collect the adequate input-output sequences and estimate the proper model because the dimension of state vector increases drastically. Therefore, the learning (observing) agent applies the CVA method to each (observed) agent separately.

3.1.2 Determine the dimension of other agent

It is important to decide the dimension n of the state vector $\boldsymbol{x}$ and lag operator l that tells how long the historical information is related in determining the size of the state vector when we apply CVA to the classification of agents. Although the estimation is improved if l is larger and larger, much more historical information is necessary. However, it is desirable that l is as small as possible with respect to the memory size. For n, complex behaviors of other agents can be captured by choosing the order n high enough.

In order to determine n, we apply Akaike's Information Criterion (AIC) which is widely used in the field of time series analysis. AIC is a method for balancing precision and computation (the number of parameters). Let the prediction error be $\boldsymbol{\varepsilon}$ and covariance matrix of error be

$$\hat{\boldsymbol{R}} = \frac{1}{N-k-l+1} \sum_{t=l+1}^{N-k+1} \boldsymbol{\varepsilon}(t)\boldsymbol{\varepsilon}^T(t).$$

Then $AIC(n)$ is calculated by

$$AIC(n) = (N-k-l+1)\log|\hat{\boldsymbol{R}}| + 2\lambda(n), \tag{6}$$

where λ is the number of the parameters. The optimal dimension n^* is defined as

$$n^* = \arg\min AIC(n).$$

Since the reinforcement learning algorithm is applied to the result of the estimated state vector to cope with the non-linearity and the error of modeling, the learning agent does not have to construct the *strict* local predict model. However, the parameter l is not under the influence of the $AIC(n)$. Therefore, we utilize $\log|\hat{\boldsymbol{R}}|$ to determine l.

1. Memorize the q dimensional vector $\boldsymbol{y}(t)$ about the agent and m dimensional vector $\boldsymbol{u}(t)$ as a motor command.

2. From $l = 1 \cdots$, identify the obtained data.

   (a) If $\log|\hat{\boldsymbol{R}}| < 0$, stop the procedure and determine n based on $AIC(n)$,

   (b) else, increment l until the condition (a) is satisfied or $AIC(n)$ does not decrease.

3.2 Reinforcement Learning

After estimating the state space model given by Eq. 2, the agent begins to learn behaviors using a reinforcement learning method. Q learning [17] 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 $\boldsymbol{\mu}(t)$ is determined, and the successive state is predicted. Therefore, we can regard an environment as Markovian.

3.3 Task and Assumptions

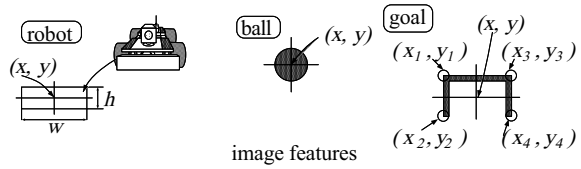
We apply the proposed method to a simple soccer-like game including two agents (Figure 12). Each agent has a single color TV camera and does not know the location, the size and the weight of the ball, the other agent, any camera parameters such as focal length and tilt angle, or kinematics/dynamics of itself. They move around using a 4-wheel steering system. 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

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

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

The output (observed) vectors are shown in Figure 13. As a result, the dimension of the observed vector about the ball, the goal, and the other robot are 4, 11, and 5 respectively.

Figure 12: The environment and our mobile robot



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

Figure 13: Image features of the ball, goal, and agent

3.4 Experimental Results

3.4.1 *Simulation Results*

Table 1: The estimated dimension (computer simulation)

agent	l	n	$\log \lvert \boldsymbol{R} \rvert$	AIC
goal	1	2	-0.001	121
ball	2	4	0.232	138
random walk	3	6	1.22	232
move to the ball	3	6	-0.463	79

Table1 shows the result of identification. In order to predict the successive situation, $l = 1$ is sufficient for the goal, while the ball needs 2 steps. The motion of the random walk agent can not be correctly predicted as a matter of course while the move-to-the-ball agent can be identified by the same dimension of the random agent, but the prediction error is much smaller than that of random walk.

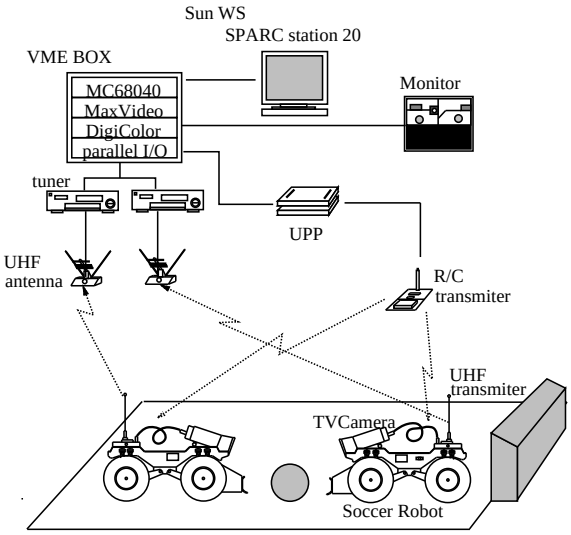


Figure 14: A configuration of the real system.

Table 2 shows the success rates of shooting and passing behaviors compared with the results in our previous work [8] in which only the current sensor information is used as a state vector. We assign a reward value 1 when the robot achieved the task, or 0 otherwise. If the learning agent uses the only current information about the ball and the goal, the leaning agent can not acquire the optimal behavior when the ball is rolling. In other words, the action value function does not become to be stable because the state and action spaces are not consistent with each other.

Table 2: Comparison between the proposed method and using current information

state vector	success of shooting (%)	success of passing (%)
current position	10.2	9.8
using CVA	78.5	53.2

3.4.2 *Real Experiments*

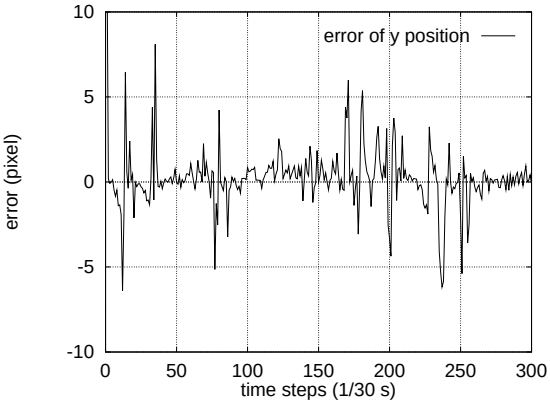
We have constructed the radio control system of the robot, following the remote-brain project by Inaba et al. [19]. Figure 14 shows a configuration of the real mobile robot system. The image taken by a TV camera mounted on the robot is transmitted to a UHF receiver and processed by Datacube MaxVideo 200, a real-time pipeline

Table 3: The estimated dimension (real environment)

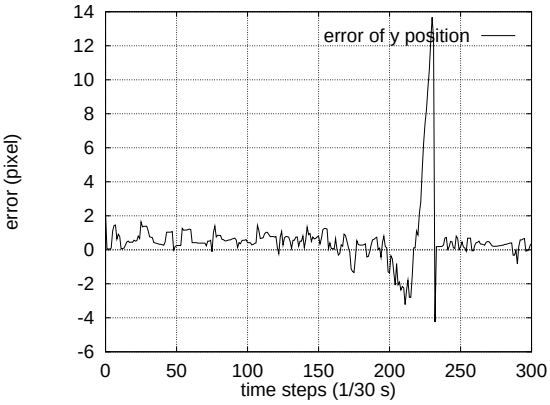
from the shooter				
	l	n	$\log \lvert \boldsymbol{R} \rvert$	AIC
ball	4	4	1.88	284
goal	1	3	-1.73	-817
passer	5	4	3.43	329
from the passer				
	l	n	$\log \lvert \boldsymbol{R} \rvert$	AIC
ball	4	4	1.36	173
shooter	5	4	2.17	284

video image processor. In order to simplify and speed up the image processing time, we painted the ball, the goal, and the opponent red, blue, and yellow, respectively. The input NTSC color video signal is first converted into HSV color components in order to make the extraction of the objects easy. The image processing and the vehicle control system are operated by VxWorks OS on MC68040 CPU which are connected with host Sun workstations via Ether net. The tilt angle is about -26 [deg] so that robot can see the environment effectively. The horizontal and vertical visual angles are about 67 [deg] and 60 [deg], respectively.

The task of the passer is to pass a ball to the shooter while the task of the shooter is to shoot a ball into the goal. Table 3 and Figure 15 show the experimental results. The value of l for the ball and the agent are bigger than that of computer simulation, because of the noise of the image processing and the dynamics of the environment due to such as the eccentricity of the centroid of the ball. Even though the local predictive model of the same ball for each agent is similar ($n = 4$, and slight difference in $\log |\boldsymbol{R}|$ and AIC) from Table3, the estimated state vectors are different from each other because there are differences in several factors such as tilt angle, the velocity of the motor and the angle of steering. We checked what happened if we replace the local predictive models between the passer and the shooter. Eventually, the large prediction errors of both side were observed. Therefore the local predictive models can not be replaced between physical agents. Figure 16 shows a sequence of images where the shooter shoots a ball which is kicked by the passer.

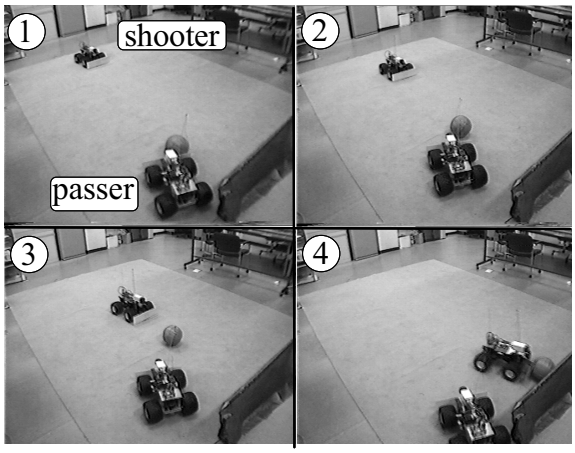


(a) y position of the ball

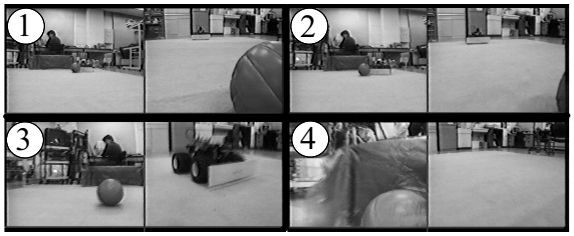


(b) y position of the left-upper of the goal

Figure 15: Prediction errors in the real environment



(a) top view



(b) obtained images (left:shooter, right:passer)

Figure 16: Acquired behavior

4 Concluding Remarks

Along with examples of soccer robots, we have claimed the importance of the design of the internal structure which reflects the complexity of the interactions with the agent's environment. Although the task and the environment seem simple and limited, the design of the soccer robots includes a variety of the fundamental and important issues as a standard problem in robotics and AI [1]. We expect that more agents in the field cause much higher interactions among them, which emerges a variety of more complex behaviors.

Acknowledgment

We like to thank Eiji Uchibe, Yasutake Takahashi, Masateru Nakamura, and Chizuko Mishima for their supports of our work described in this paper.

References

[1] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara. "robocup: A challenge problem of ai". *AI Magazine*, 18:73–85, 1997.

[2] M. Asada, S. Noda, and K. Hosoda. Action-based sensor space categorization for robot learning. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems 1996 (IROS '96)*, pages 1502–1509, 1996.

[3] Y. Takahashi, M. Asada, and K. Hosoda. Reasonable performance in less learning time by real robot based on incremental state space segmentation. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems 1996 (IROS96)*, pages 1518–1524, 1996.

[4] E. Uchibe, M. Asada, and K. Hosoda. Vision based state space construction for learning mobile robots in multi agent environments. In *Proceedings of 6-th European Workshop on Learning Robots, EWLR-6*, pages 33–41, 1997.

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

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

[7] M. Mataric. "Reward functions for accelerated learning". In *Proc. of Conf. on Machine Learning-1994*, pages 181–189, 1994.

[8] 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.

[9] Peter Stone and Manuela Veloso. Using machine learning in the soccer server. In *Proc. of IROS-96 Workshop on Robocup*, 1996.

[10] E. Uchibe, M. Asada, and K. Hosoda. Behavior coordination for a mobile robot using modular reinforcement learning. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems 1996 (IROS96)*, pages 1329–1336, 1996.

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

[12] Tuomas W. Sandholm and Robert H. Crites. On multiagent Q-learning in a semi-competitive domain. In *Workshop Notes of Adaptation and Learning in Multiagent Systems Workshop, IJCAI-95*, 1995.

[13] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8:293–321, 1992.

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

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

[16] Peter Van Overschee and Bart De Moor. A unifying theorem for three subspace system identification algorithms. *Automatica*, 31(12):1853–1864, 1995.

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

[18] E. Uchibe, M. Asada, and K. Hosoda. Behavior coordination for a mobile robot using modular reinforcement learning. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems 1996 (IROS96)*, pp. 1329–1336, 1996.

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