Vision Based State Space Construction for Learning Mobile Robots in Multi Agent Environments

Eiji Uchibe[*] **Minoru Asada**[**], **and Koh Hosoda**[**]

[*]*Dept. of Mechanical Engineering for Computer Controlled Machinery, Osaka University, Suita, Osaka 565, Japan uchibe@robotics.mech.eng.osaka-u.ac.jp*

[**]*Dept. of Mechanical Engineering for Adaptive Machine Systems, Osaka University, Suita, Osaka 565, Japan*

Abstract. State space construction is one of the most fundamental issues for the reinforcement learning methods to be applied to real robot tasks because they need a well-defined state space so that they can converge correctly. Especially in multi-agent environments, the problem becomes more difficult since visual information observed by the learning robot seems irrelevant to its self motion due to actions by other agents of which policies are unknown. This paper proposes a method which estimates the relationship between the learner's behaviors and the other agents' ones in the environment through interactions (observation and action) using the method of system identification to construct a state space in such an environment. In order to determine the state vectors of each agent, Akaike's Information Criterion is applied to the result of the system identification. Next, reinforcement learning based on the estimated state vectors is utilized to obtain the optimal behavior. The proposed method is applied to soccer playing physical agents, which learn to cope with a rolling ball and moving other agent. The computer simulations and the real experiments are shown and a discussion is given.

1 Introduction

Building a robot that learns to perform a task through visual information has been acknowledged as one of the major challenges facing Vision, 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 a multi-agent environment, the conventional reinforcement learning algorithm does not seem applicable because the environment including the other learning agents seems to change randomly from a viewpoints of the learning agent. There are two major reasons why the learning would be difficult in a multi-agent environment.

1. The other agent may use a stochastic action selector which could take a different action even if the same sensation occurs to it.
2. The other agent has a perception (sensation) different from the learning agent's. This means that the learning agent would not be able to discriminate different situations which the other agent can do, and vice versa.

Therefore, the learner cannot predict the other agent behaviors correctly even if its policy is fixed 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.

Littman (Littman, 1994) proposed the framework of Markov Games in which Q-learning agents try to learn a mixed strategy optimal against the worst possible opponent in a zero-sum 2-player game in a grid world. He assumed that the opponent's goal is given to the learner (opponent tries to minimize a single reward function, while it is to be maximized by the learning agent). Sandholm and Crites (Sandholm and Crites, 1995) studied the ability of a variety of Q-learning agents to play iterated prisoner's dilemma game against an unknown opponent. They showed that adequate previous moves and sensations are needed for the successful learning. Lin (Lin and Mitchell, 1992) compared recurrent-Q based on a recurrent network with window-Q based on both the current sensation and the N most recent sensations and actions, and he showed the former is superior to the latter because a recurrent network can cope with historical features appropriately. However, it is still difficult to determine the number of neurons and the structures of network in advance. Furthermore, their methods utilize the global information. Although the uncertainties of sensor and actuator outputs are considered by a stochastic transition model in the state space, such a model cannot account for the accumulation of sensor errors in estimating the robot position. Further, from the viewpoint of real robot applications, we should construct the state space so that it can reflect the outputs of the physical sensors which are currently available and can be mounted on the robot.

Robotic soccer is a good domain for studying multi-agent problems (Kitano et al., 1997). Stone and Veloso proposed *layered learning* method which consists of two levels of learned behaviors (Stone and Veloso, 1996). The lower is for basic skills such as interception of a moving ball and the higher is one which can make decisions whether or not to make a pass using decision tree. Uchibe et al. proposed a method of modular reinforcement learning which coordinates multiple behaviors taking account of a trade-off between learning time and performance (Uchibe et al., 1996). Since these methods utilize the current sensor outputs as states, their methods can not cope with the motions of objects.

As described above, the existing methods in multi agent environments need state vectors in order fort them 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.

In this paper, we propose 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) (Akaike, 1974) to the result of Canonical Variate Analysis(CVA) (Larimore, 1990), 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 opponent), 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. The simulation results and the real experiments are shown and a discussion is given.

2 Construction of the internal state model from observation and action

2.1 Local predictive models for other agents

In order to succeed in learning, it is necessary for the learner to predict the successive situations as mentioned above. However, the agent can not obtain the complete information necessary to correctly predict them because of *partial observation* due to the limitation of sensing capability. Since we consider that the robot should construct the state space from its viewpoint, what the learning agent can do is to collect all the observed data with the motor commands taken during the observation and to estimate the relationship between the observed agents and the learner's behaviors in order to take an adequate behavior although it might not be guaranteed to be optimal. In the following, we consider to utilize a method of system identification, regarding the previous observed data and the motor commands as the inputs, and future observation results as the outputs of the system, respectively.

Figure 1 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

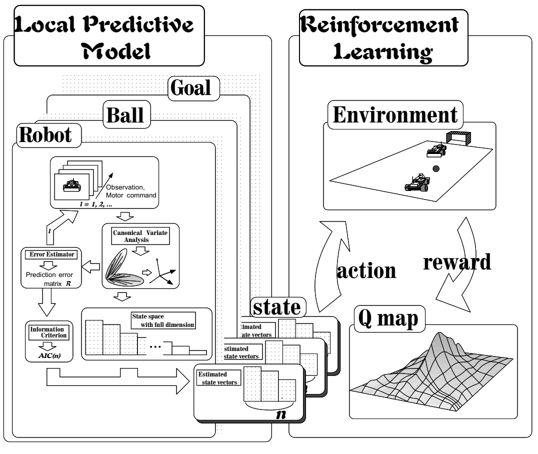


Figure 1: An overview of the proposed method

predictive models, which are described in section 2.2. By approximating the relationship between inputs (learner's action) and 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 (Section 2.2).

2. Estimate the state space with the full dimension directly from the observations and motor commands (Section 2.2).

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 (Section 2.3).

4. Apply the reinforcement learning based on the estimated state vectors (Section 3).

2.2 Canonical Variate Analysis

A number of algorithms to identify multi-input multi-output (MIMO) combined deterministic-stochastic systems have been proposed (Van Overschee and De Moor, 1995). 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) (Larimore, 1990) is one of such algorithms, which uses canonical correlation analysis to construct a state estimator. We define $\boldsymbol{P}$ and $\boldsymbol{F}$ as follows : the past inputs and outputs

$$\boldsymbol{P} := \left(\begin{array}{ccc} \boldsymbol{p}(1) & \cdots & \boldsymbol{p}(N/2-1) \end{array} \right),$$

the future outputs

$$\boldsymbol{F} := \left(\begin{array}{ccc} \boldsymbol{f}(N/2) & \cdots & \boldsymbol{f}(N) \end{array} \right),$$

and we denote the future input block Hankel matrix as $\boldsymbol{U}$. CVA algorithm is insensitive to scaling of the inputs (motor commands) and/or the outputs (sensor outputs), because the CVA algorithm considers only the angles and the normalized directions between the past inputs and outputs orthogonalized to the future inputs ($\boldsymbol{P}/\boldsymbol{U}^{\perp}$) and the future outputs orthogonalized to the future inputs ($\boldsymbol{F}/\boldsymbol{U}^{\perp}$) (Van Overschee and De Moor, 1995), where $\boldsymbol{A}^{\perp}$ denotes the subspace perpendicular to the row space of $\boldsymbol{A}$, and $\boldsymbol{B}/\boldsymbol{A}$ is shorthand for the projection of the row space of $\boldsymbol{B}$ onto the row space of $\boldsymbol{A}$ (See Figure 2).

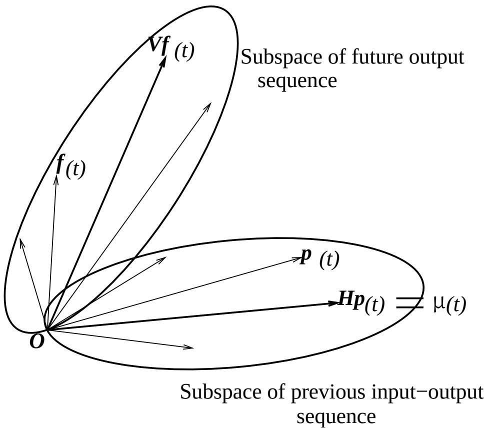

Vf (t)
Subspace of future output sequence
f(t)
p (t)
Hp(t)
μ(t)
O
Subspace of previous input−output sequence

Figure 2: the subspace

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

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

with

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

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. Larimore's Canonical Variate Analysis (CVA) (Larimore, 1990) is one of identification algorithms, which uses canonical correlation analysis to construct a state estimator. 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

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

2. Compute the 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. Apply singular value decomposition

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

and $\boldsymbol{U}$ and $\boldsymbol{V}$ are calculated as:

$$\begin{aligned} \boldsymbol{U} &:= \boldsymbol{U}_{aux}^T\hat{\boldsymbol{\Sigma}}_{pp}^{-1/2}, \\ \boldsymbol{V} &:= \boldsymbol{V}_{aux}^T\hat{\boldsymbol{\Sigma}}_{ff}^{-1/2}. \end{aligned}$$

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

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

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

Strictly speaking, the learning agent should construct the local predictive model about the whole system since all the agents do in fact interact. However, it is intractable to collect the adequate input-output sequences and estimate the proper model because the dimension of state vector drastically increases. Therefore, the learning (observing) agent obtains the local predictive models by applying the CVA method to all the (observed) agents separately.

2.3 Determination of the dimension of other agent

It is important to decide the dimension n of the state vector $\boldsymbol{\mu}$ and lag operator l that tells how long the historical information is involved 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, but we have to take account of the trade off between the number of parameters and the precision of the estimation.

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

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

where

$$\lambda(n) = n(2p+m) + pm + \frac{1}{2}p(p+1). \quad (7)$$

The optimal dimension n^* is defined as

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

where

$$1 \le n \le \min(l(m+q), kq).$$

However, the parameter l is not under the influence of the $AIC(n)$. Because the reinforcement learning algorithm is applied to the result of the estimated state vector in order to cope with the non-linearity and the error of modeling, the learning agent does not have to construct the *strict* local predict model. 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 Reinforcement Learning

After estimating the state space model represented by Eq. 2, the agent begins to learn behaviors using a reinforcement learning method. Q learning (Watkins and Dayan, 1992) is a form of model-free reinforcement learning based on the stochastic dynamic programming. It provides robots with the capability of learning to act optimally in a Markovian environment. In the previous section, the appropriate dimension n of the state vector $\boldsymbol{\mu}(t)$ has been determined, and the successive state can be predicted. Therefore, we regard an environment as Markovian.

In order to utilize the result of identification for the Q learning, the state vector $\boldsymbol{\mu}$ has to be quantized. Because the state vector $\boldsymbol{\mu}$ is calculated as

$$E\{\boldsymbol{\mu}\boldsymbol{\mu}^T\} = \boldsymbol{I}_n,$$

we segment the state space as

$$\mu_i < -1, \quad -1 \le \mu_i < 1 \quad 1 \le \mu_i, \quad \text{for all } i.$$

Hereafter, we denote the estimated state vector $\boldsymbol{\mu}$ as $\boldsymbol{x}$ for reader's understanding. We assume that the robot can discriminate the set $\boldsymbol{X}$ of distinct world states, and can take the set $\boldsymbol{U}$ of actions on the world. A simple version of a Q learning algorithm used here 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 action value function.
4. Carry out action u in the environment. Let the next state be x' and immediate reward be r.
5. Update action value function from x, u, x', and r,

$$\begin{aligned} Q_{t+1}(x,u) &= (1-\alpha_t)Q_t(x,u) \\ &+ \alpha_t(r + \gamma \max_{u' \in \boldsymbol{U}} Q_t(x', u')) \quad (8) \end{aligned}$$

   where α_t is a learning rate parameter and γ is a fixed discounting factor between 0 and 1.
6. Return to 2.

4 Task and Assumptions

We apply the proposed method to a simple soccer-like game including two agents (Figure 3). Each agent has a single color TV camera and does not know the locations, the sizes and the weights of the ball and 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. The effects of an action against the environment can be informed to the agent only through the visual information. 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

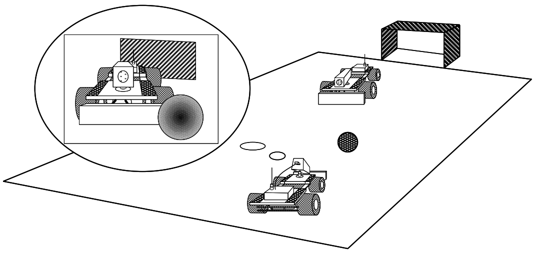


Figure 3: The environment and our mobile robot

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

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 4. In case of the ball, the center position of the ball image (x_c, y_c) is used, and the both ends (x_l, y_l) and (x_r, y_r) are used for the field lines. In case of the goal, the four corners of the goal image (x_{ul}, y_{ul}), (x_{bl}, y_{bl}), (x_{ur}, y_{ur}), and (x_{br}, y_{br}) are used. In case of other agent, the center of position, the width and the height of the plate are used. As a result, the dimension of the observed vector for

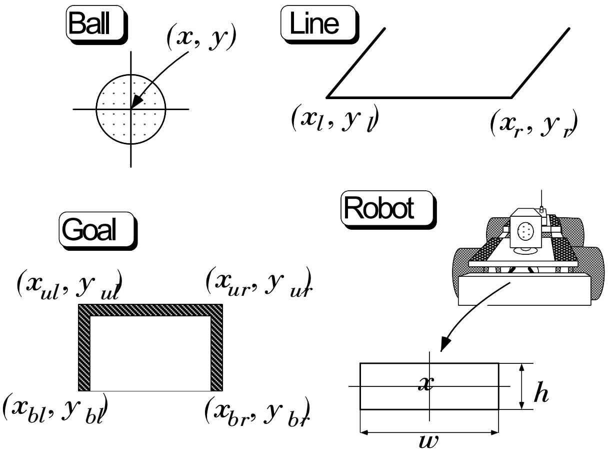


Figure 4: Image feature points of the ball, goal, line and agent

the ball, the goal, the line, and the agent are 2, 4, 8, and 3 respectively.

We assume that the other active agent has some basic behaviors designed by programmer such that 1) to move randomly, or 2) to move to the ball, and that it does not change its behavior frequently.

5 Experimental Results

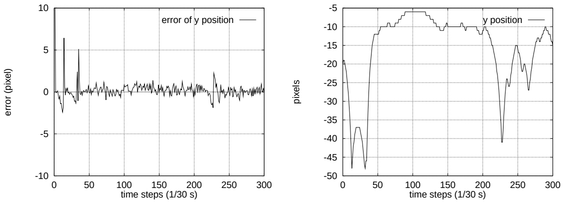
5.1 Simulation Results

Table 1 shows the result of identification. In order to predict the successive situations $l = 1$ is sufficient for the goal and line, while the ball needs 2 steps. Figures 5 show the result (error and trajectory) of the ball. At time steps 40 and 230, the learner kicked the ball, therefore prediction errors become large drastically.

Table 1: The estimated dimensions (computer simulation)

agent	l	n	$\log \lvert \boldsymbol{R} \rvert$	AIC
line	1	3	-2.14	-800
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

The observing agent can not predict the random walk agent as a matter of course. The agent which moves to the ball can be identified by the same dimension of the random agent, but the prediction error is much smaller than that of the random walk.



(a) prediction error

(b) real trajectory

Figure 5: Prediction errors of the y position of the ball

5.1.1 Shooting behavior

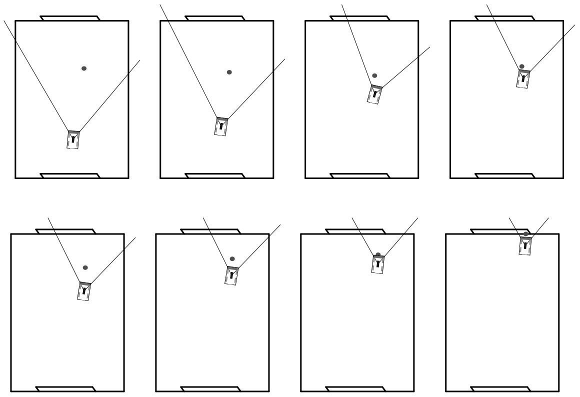


Figure 6: The robot succeeded in shooting a moving ball into a goal

Figure 6 shows a sequence of shooting a slowly moving ball into the goal using CVA method and Table 2 shows a comparison about the success rate of shooting with the model based on only the current perception and action. We assign a reward value 1 when the ball was kicked into the goal or 0 otherwise and the environment consists of the ball, the goal and the line. The two lines emerged from the agent show its visual angle.

If the learning agent uses the only current observation as the state vectors 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.

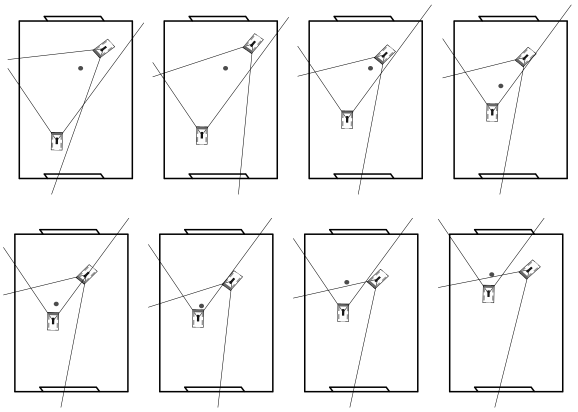


Figure 7: The robot succeeded in passing a ball to the other agent

5.1.2 Passing behavior

Passing a ball to the other agent is regarded as shooting (kicking) a ball toward the moving goal. We assign a reward value 1 when the ball was kicked into the other agent, -0.8 when the learner makes a collision with the other agent or 0 otherwise and the environment consists of the ball and the other agent.

Table 2: Comparison between the proposed method and the other one using only the current observation

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

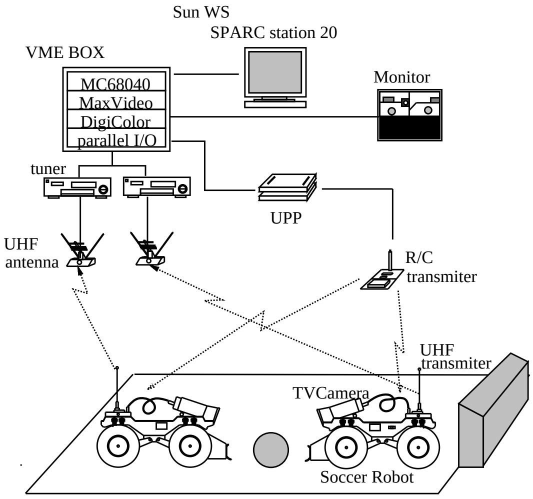
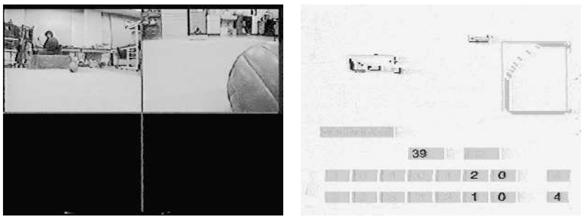


Figure 8: A configuration of the real system.



(a) input image (left : shooter, and right : passer)

(b) feature extraction

Figure 9: Detection of the agents

5.2 Real Experiments

We have constructed the radio control system of the robot, following the remote-brain project by Inaba et al. (Inaba, 1993). Figure 8 shows a configuration of the real mobile robot system. The image taken by a TV camera mounted on each 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, the goal, and the opponent in red, blue, and yellow, respectively. The input NTSC color video signal is first converted into HSV color components in order to easily extract the objects. Figure 9(a) and (b) show the images taken by a TV camera mounted on each robot (left : shooter, and right : passer). 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.

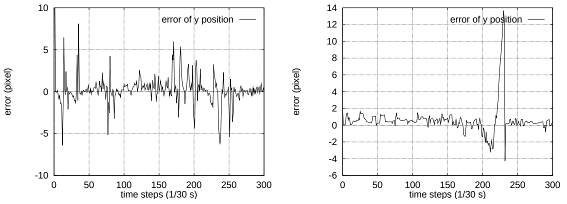
The task of the passer is to pass a ball to the shooter while the task of the shooter is to shoot a ball into the goal. Table 3 and Figure 10 show the experimental results. The value of l for the ball and the agent are bigger than that of computer simulation because of the noise of the image processing and the dynamics of the environment 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) (See Table3), the estimated state

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

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. Finally, Figure 11 shows a sequence of images where the passer kicked a ball towards the shooter, which shot it into the goal.



(a) y position of the ball

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

Figure 10: Prediction error in real environment

6 Conclusion

This paper presents a method of behavior acquisition in order to apply reinforcement learning to the environment including other agents. Our method takes account of the trade-off among the precision of prediction, the dimension of state vector, and the length of steps to identify the model. Our robots can shoot and pass a ball even if a ball is rolling well.

As future work we plan to challenge the following

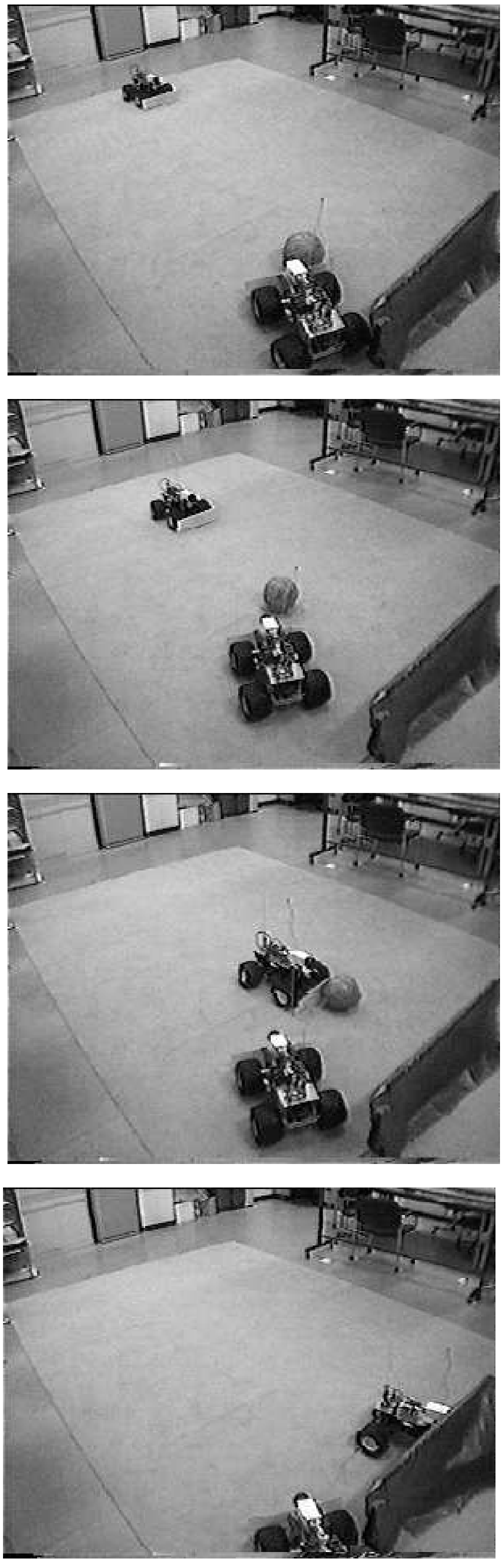


Figure 11: Acquired behavior

issues:

- The local predictive model provided the state vectors by which prediction can be effectively done because they have strongly correlation between the past inputs/outputs and the future outputs. In order to accomplish the more complicated task, the learning robot can determine the minimum dimension of the state vectors in accordance with the increase of the level of the task complexity (Uchibe and Asada, 1997). We are planning to extend the "Learning from Easy Missions" paradigm (Asada et al., 1995) to the complicated task.

- In our experiments, we quantized each of the elements of the estimated state vectors into three categories based on its variance. Several segmentation methods such as Parti game algorithm (Moore and Atkeson, 1995) and Asada's method (Asada et al., 1996) can be alternative.

- In order to accomplish the more complicated cooperative tasks, the learning robot should estimates the interaction among all the objects.

References

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

Asada, M., Noda, S., and Hosoda, K. (1996). Action-based sensor space categorization for robot learning. In *Proc. of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems.*

Asada, M., Noda, S., Tawaratsumida, S., and Hosoda, K. (1995). Vision-based reinforcement learning for purposive behavior acquisition. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 146–153.

Connel, J. H. and Mahadevan, S. (1993). *Robot Learning.* Kluwer Academic Publishers.

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

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

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

Lin, L.-J. and Mitchell, T. M. (1992). Reinforcement learning with hidden states. In *Proc. of the 2nd International Conference on Simulation of Adaptive Behavior: From Animals to Animats 2.*, pages 271–280.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Proc. of the 11th International Conference on Machine Learning*, pages 157–163.

Moore, A. W. and Atkeson, C. G. (1995). The partigame algorithm for variable resolution reinforcement learning in multidimensional state-spaces. *Machine Learning*, 21:199–233.

Sandholm, T. W. and Crites, R. H. (1995). On multiagent Q-learning in a semi-competitive domain. In *Workshop Notes of Adaptation and Learning in Multiagent Systems Workshop, IJCAI-95.*

Stone, P. and Veloso, M. (1996). Using machine learning in the soccer server. In *Proc. of IROS-96 Workshop on Robocup.*

Uchibe, E. and Asada, M. (1997). Learning and development for physical animats: Environmental complexity control for vision-based mobile robot. In *Fourth European Conference on Artificial Life (ECAL97).*

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

Van Overschee, P. and De Moor, B. (1995). A unifying theorem for three subspace system identification algorithms. *Automatica*, 31(12):1853–1864.

Watkins, C. J. C. H. and Dayan, P. (1992). Technical note: Q-learning. *Machine Learning*, pages 279–292.