# Action-Based Sensor Space Categorization for Robot Learning

Minoru Asada, Shoichi Noda*, and Koh Hosoda
Dept. of Mech. Eng. for Computer-Controlled Machinery
Osaka University, 2-1, Yamadaoka, Suita, Osaka 565, Japan
asada@robotics.ccm.eng.osaka-u.ac.jp

**Abstract**

*Robot learning such as reinforcement learning generally needs a well-defined state space in order to converge. However, to build such a state space is one of the main issues of the robot learning because of the inter-dependence between state and action spaces, which resembles to the well known "chicken and egg" problem. This paper proposes a method of action-based state space construction for vision-based mobile robots. Basic ideas to cope with the inter-dependence are that we define a state as a cluster of 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 that this sequence is defined as one action. To realize these ideas, we need many data (experiences) of the robot and cluster the input vectors as hyper ellipsoids so that the whole state space is segmented into a state transition map in terms of action from which the optimal action sequence is obtained. To show the validity of the method, we apply it to a soccer robot which tries to shoot a ball into a goal. The simulation and real experiments are shown.*

## 1 Introduction

Building a robot that learns to perform a task has been acknowledged as one of the major challenges facing Robotics and AI [1]. Recently, reinforcement learning [2, 3] and memory-based learning [1] have 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 these 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 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 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 [4].

Generally, the design of the state space in which necessary and sufficient information to accomplish a given task is included depends on the capability of agent actions. On the other hand, the design of the action space also depends on the capability of perception. This resembles the well-known "chicken and egg problem" that is difficult to be solved (see Figure 1).

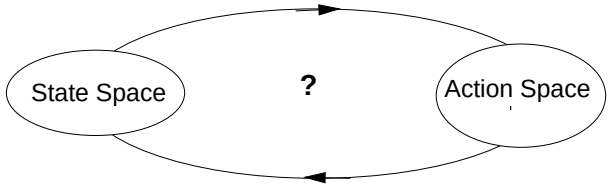



Figure 1: The inter-dependence between state and action spaces

One can construct a state space fixing the action space first. Chapman and Kaelbring [5] proposed an input generalization method which splits an input vector consisting of a bit sequence of the states based on the already structured actions such as "shoot a ghost" and "avoid an obstacle." However, the original states have been already abstracted, and therefore it seems difficult to be applied to the continuous raw sensor space of real world.

Dubrawski and Reingnier [6], and Kröse and Dam[7] proposed methods similar to each other which abstracted sonar information into the form useful for mobile robots to avoid obstacles. Ishiguro et al. [8] dealt with a problem of state space categorization by statistically analyzing the sensor patterns, actions, and rewards given at the end of goal achievement. Since they deal with reflexive behaviors such as obstacle avoidance, these methods do not suffer from the fixed length physical actions. However, in case of a task to achieve the goal farther from the viewpoint based

*Currently, he is with Hitachi Co.

on the visual information, the same physical actions might cause different changes in image, and therefore it seems difficult to specify the state and action spaces by which learning converges correctly [1].

Asada et al. [9] called this "a state-action deviation problem" due to the difference in resolution between the robot action in a 3-D space and the projection of its effect onto a 2-D image. They have given one solution for this problem by restructuring the action space so that one action may cause one state transition. That is, first they divided the sensor space by hand, and then constructed the action space so that the sensor and action spaces can be consistent with each other. However, there is no guarantee that such a state space is always appropriate for the robot.

This paper propose a method that recursively splits an input vector from the sensor based on a definition of action primitive that is a resultant action caused by a motor command executed during the fixed time interval. The basic ideas are as follows:
We define

1. a state as a set of input vectors from which the robot achieves the goal or already acquired state by a variable sequence of one kind action primitive, and

2. an action as a sequence of action primitive that causes a state transition.

Recently, a study on mechanisms for emergent machine intelligence through the interaction with agents' environment has been receiving increased attention [10]. The proposed method in this paper can be regarded as one that solves the so-called "segmentation" problem through the interactions with the agent's environment. That is, the designer does not provide the state and action spaces needed to accomplish a given task from his or her viewpoint. Instead, the robot constructs the necessary state and action spaces based on its experiences (interactions with its environment). The construction process corresponds to behavior learning, and as a result the purposive behavior is acquired.

The remainder of this article is structured as follows: In the next section, we describe our method to automatically construct the sensor and action spaces with a simple computer simulation. Then, we show the results of the experiments of a vision-based mobile robot that tries to shoot a ball into a goal. Finally, we give our motivation to the approach described in this paper and our future work.

# 2 Sensor and Action Space Construction

As we described in the above, the state space[2] designed by programmer is not always appropriate for the robot to accomplish a given task. If multiple states to be discriminated from each other are categorized into the same state, the distribution of that state transitions widely spreads out, and therefore it seems difficult for the robot to achieve the goal. On the other hand, if the size of the state space is too large due to unnecessary separations, the learning incredibly takes long time (it is generally an exponential order in the size of the state space [11]). Then, we attempt at solving this problem by making the robot construct the state space, that is, it should find a state space by itself through interactions with the environment. The following are the requirements for the problem:

1. The state and action spaces should reflect physical sensor(s) and actuator(s) of a robot. The 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) are useful only for simple toy problems in computer simulations.

2. Since everything changes asynchronously in real world [4], the state and action spaces directly reflecting the physical sensors and actuators suffer from the state - action deviation problem. The state and action spaces should be restructured to cope with this problem.

3. The sensor space categorization should be robust against the various disturbances such as sensor noise, delay, and uncertainty of action execution.

## 2.1 The Method

Basic ideas of our method are that we define:

1. an action primitive $a_i$ $(i = 1, 2, ..., n)$ as a resultant action caused by a motor command executed during the fixed time interval,

2. a state as a set of input vectors from which the robot achieves the goal or already acquired state by a variable sequence of one kind action primitive, and

3. an action as a sequence of action primitive that causes a state transition,

and that such states are found in the order of closeness to the goal state.

Figure 2 shows a state space in terms of the goal state and actions. $S_1^i$, $i = a_1, a_2, a_3, ..., a_n \in \mathbf{A}$ (a set of action primitives) and $\mathbf{S_1}$ denote a state from which the robot can achieve the goal by only one action $i$, and a set of these states, respectively. Further, $\mathbf{S_2}$ denotes a set of states from each of which the robot can achieve $\mathbf{S_1}$ only by one action. Similarly, $\mathbf{S_j}$ denotes a set of states from which the robot can achieve the goal at least $j$ actions. Any input vector from which the robot can achieve the goal can be included in any state in $\mathbf{S_k}$ $(k = 1, 2, .., j)$. The algorithm to obtain such a state space is given below.

[1] In case of vision sensors, the same action might cause large change in image if the object is close to the observer, and small change if it is farther.

[2] Here, we suppose the state space is a space consisting of input vector from sensors. In control theory, this is not always true.

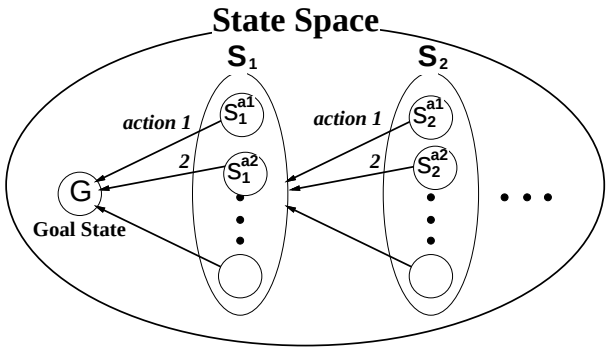



Figure 2: The goal-directed and action-based state space construction

**Algorithm**

1. Set the goal state as a target zone $\boldsymbol{Z}_T$.
2. Take an action randomly. From the definition, the same action primitive is iteratively executed until the robot achieves the target zone or the fixed time interval expires.
3. Store an input vector $\boldsymbol{x} \in \boldsymbol{R}^m$ (m: the size of the vector) with an index of the action $a \in \boldsymbol{A}$ the robot took when it could succeed in achieving $\boldsymbol{Z}_T$ from $\boldsymbol{x}$. Do not include the vectors that have been already categorized into the previously divided states.
4. Fit a multi-dimensional uniform distribution function (a concentration ellipsoid [12]) to a cluster of stored vectors with the same action index $a_i \in \boldsymbol{A}$ obtained above, and construct a state $s_{a_i}$ ($a_i \in \boldsymbol{A}$). The boundary surface of the ellipsoid is given by:

$$(\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) = m + 2, \qquad (1)$$

   where $\mu$ and $\Sigma$ denote the mean vector and the covariance matrix, respectively.
5. Update the target zone $\boldsymbol{Z}_T$ as a union of states $s_{a_i}$ ($i = 1, 2, .., n$) obtained in the previous step. If a vector is categorized into plural states $s_{a_j}$ ($j = 1, ..$) (clusters are overlapped), select one state so that the following distance normalized by its covariance can be the minimum:

$$\Delta_j = (\boldsymbol{x} - \boldsymbol{\mu}_j)^T \Sigma_j^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_j)$$

6. Stop if the state space is almost covered by the divided clusters. Else, go to 2.

We call a set of states $s_a$ ($a \in \boldsymbol{A}$) the i-th closest to the goal state $\mathbf{S_i}$. By the above algorithm, we can obtain not only the sensor space categorization but also the optimal path to the goal from everywhere.

# 3 Experimental Results and Remarks

## 3.1 Simulation (I)

To show the validity of the proposed method, we show a simple computer simulation in toy world consisting of $100 \times 100$ grids. The task for the robot is to enter the circle area whose radius is 5, located at the center of the world. The action primitives are 1.0 grid motion into any of four directions (up, down, left, and right). The input vector is an absolute coordinate $(x, y)$ (real number) of the robot location.

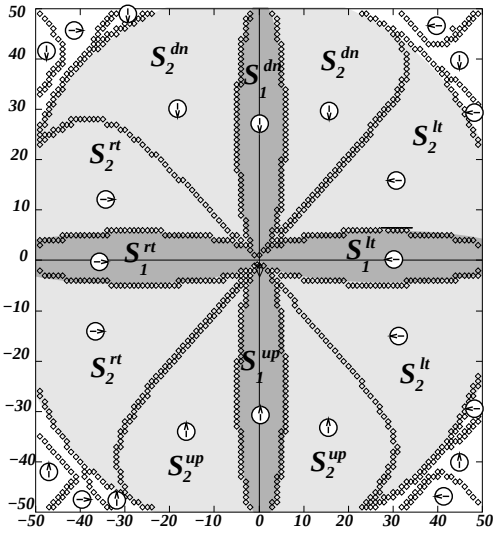



Figure 3: Result of the variable action-based construction of 2D state space

Figure 3 shows a result where the whole state space is separated into twelve states. The states obtained earlier have darker colors. Each arrow indicates an action by which the state is classified. A set of states $\mathbf{S_1}$ from which the robot can achieve the goal by only one action consists of $S_1^{up}$, $S_1^{dn}$, $S_1^{lt}$, and $S_1^{rt}$ corresponding to upward motion and so on. A set of states $\mathbf{S_2}$ from which the robot can achieve $\mathbf{S_1}$ by only one action consists of four ellipsoids $S_2^{up}$, $S_2^{dn}$, $S_2^{lt}$, and $S_2^{rt}$ which are projected onto eight regions in Figure 3.

This figure can be interpreted as follows: the top-left region $S_2^{dn}$ indicates downward motion, and the robot takes this action. Then, the robot seems to enter a state $S_2^{rt}$ of which action is rightward. However, this state is classified into the same $\mathbf{S_2}$, therefore, the robot continues to take the same action (downward) until entering the state $S_1^{rt}$ belonging to $\mathbf{S_1}$, and state transition occurs[3] This means that the same input

[3]Even if the robot changes its actions when it crosses the state boundary between $S_2^{dn}$ and $S_2^{rt}$, the consequent actions are iterations of downward and leftward ones, and finally it

vector, the absolute coordinate of the robot location here, might be different depending on its action. We sometimes have a similar situation that we failed to recognize the intersection which we usually cross over from the certain direction when we cross it from the different direction.

The top-left quarter in the figure can be originally either downward or leftward motion, and accidentally classified into one by selecting the shorter one of the distances from the center of two ellipsoids.

Let us consider to apply the existing methods such as [6, 7] to the same problem. Since the length of one action is fixed in these methods, the size of the obtained state space depends on the length of one action and/or the size of the world. However, the size of the state space would not be so affected by these parameters in our method since the length of one action can be variable. This suggests that if the topology of an environment is similar to another, the state spaces obtained by our method are not so different regardless of the absolute size of the environment.

## 3.2 Simulation (II)

### 3.2.1 Task and environment

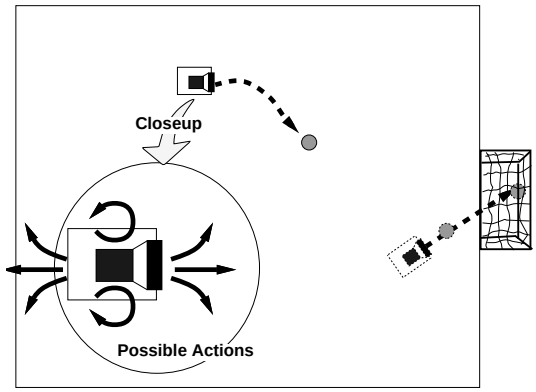



Figure 4: Task

As a more complicated task for the robot, we consider an environment shown in Figure 4 where the task for a mobile robot is to shoot a ball into a goal (same as in [9]). The environment consists of a ball and a goal, and the mobile robot has a single TV camera. The robot does not know the location and the size of the goal, the size and the weight of the ball, any camera parameters such as focal length and tilt angle, or kinematics/dynamics of itself.

We performed the computer simulation with the following specifications. The field is a square of 3.0m × 3.0m. The goal post is located at the center of the top line of the square (see Figure 4) and its height and width are 0.23m and 0.9m, respectively. The robot is 0.31m wide and 0.45m long and kicks a ball of diameter 0.09m. The maximum translation velocity is 1.1m/s, and the maximum angular velocity is 4.8 rad/s. The camera is horizontally mounted on the robot (no tilt), and its visual angle is 36 degrees. The velocities of the ball before and after being kicked by the robot is calculated by assuming that the mass of the ball is negligible compared to that of the robot. The speed of the ball is temporally decreased by a factor 0.8 in order to reflect the so-called "viscous friction." The values of these parameters are determined so that they can roughly simulate the real world.

The robot can select an action to be taken in the current state of the environment. The robot moves around using a PWS (Power Wheeled Steering) system with two independent motors. Since we can send the motor control command to each of the two motors separately, we construct the action primitives in terms of two motor commands $\omega_l$ and $\omega_r$, each of which has 3 sub-actions, forward, stop, and back. All together, we have 9 actions in the action primitive set $\boldsymbol{A}$. Actually, a stop motion does not causes any changes in the environment, we do not select this action primitive.

In computer simulation, we take into account two sources of disturbances which make the method unstable. They are delays due to sensory information processing and uncertainty of action execution. The contents of the image processing are color filtering (a ball and a goal are painted in red and blue, respectively), edge enhancement, localizing and counting edge points, and vector calculation [9]. We have been using a pipeline image processor for the real robot experiments and it takes about 33 ms to perform these processes, that is, a period of one action primitive. The latter is caused by the delay necessary to stabilize motor rotation after sending motor commands, and it is about 100 ms. Therefore, the uncertainty of the action execution increases when motor commands often change.

### 3.2.2 Results

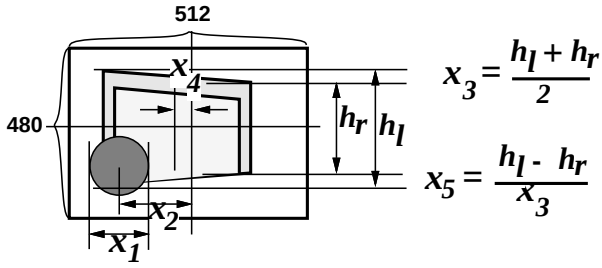
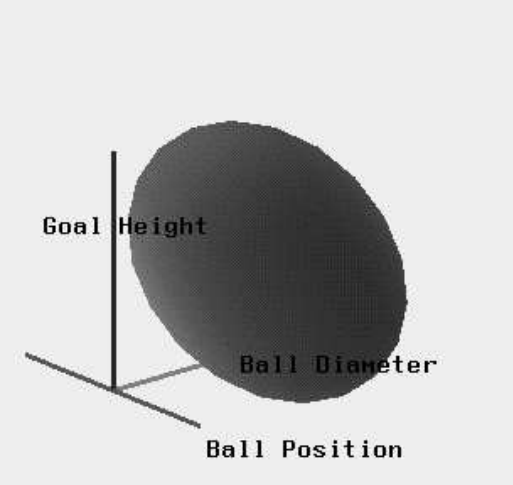



Figure 5: Input vector consisting of five parameters

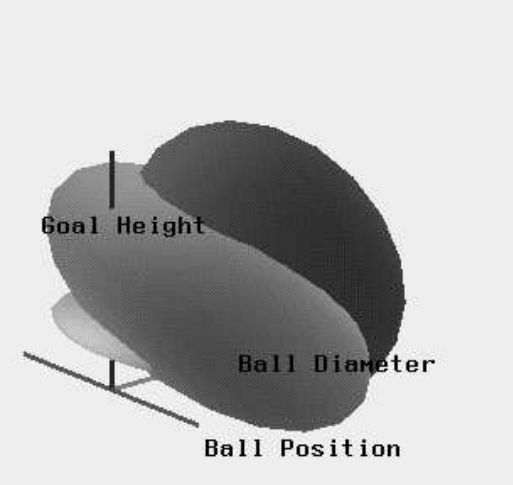
The size of the observed image is 512 by 480 pixels, and the center of image is the origin of the image coordinate system (see Figure 5). An input vector $\boldsymbol{x}$ for a shooting task consists of:

- $x_1$: the size of the ball, the diameter that ranges from 0 to about 270 pixels,

achieves the goal state with physically the same steps.




(a) the first stage ($S_1^F = \mathbf{S_1}$)




(b) the second stage ($\mathbf{S_1} + \mathbf{S_2}$)

Figure 6: Construction process of the state space for the shooting robot

- $x_2$: the position of the ball ranging from -270 to +270, considering the partial observation,
- $x_3$: 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.

As a result of the state space construction, we can obtain the state transition graph simultaneously. Actually, we have other states than obtained by the method such as "only a ball is observed" or "only a goal is observed." State transitions from these states to the obtained states is possible if they can be realized by only one action. Otherwise, it seems difficult to find a path because the robot might have many hidden states during the desirable state transition. For example, the robot exists between a ball and a goal, and the robot must take a circular motion so that it can get a position from where the ball and the goal can be observed simultaneously. During such a motion, the robot might have many hidden states. We do not deal with the "hidden states" problem here.

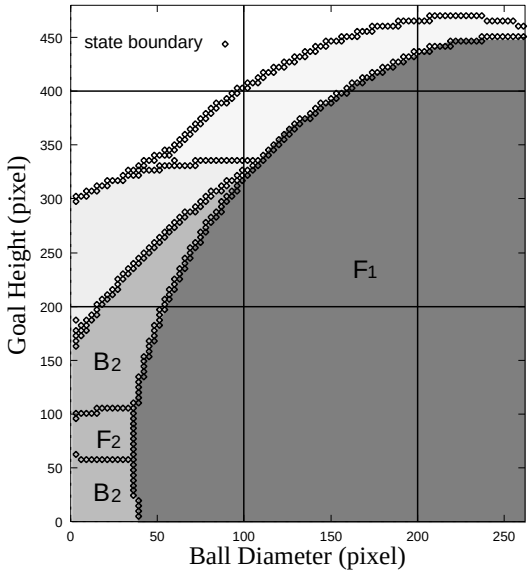



Figure 7: 2-D projection of the result of state space construction

Figure 6 shows the process of state space division. The state space in terms of ball size, ball position, and goal size is indicated when the position and the orientation of the goal ($x_4$ and $x_5$) are both zeros (in front of the goal). In the first step, only one big ellipsoid ($\mathbf{S_1^F}$) is obtained that corresponds to the forward motion (Figure 6 (a)). In the second step, two ellipsoids ($S_2^F$ and $S_2^B$) corresponding to forward and backward motions, respectively, are obtained and they construct ($\mathbf{S_2}$) (Figure 6 (b)).

For the sake of readers understanding, Figure 7 shows the projected map of the final result onto the ball-size and goal-size space when other parameters are all zeros. 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 in-

dicate the boundaries divided by programmer in the previous work [9]. The remainder of the state space in Figure 7 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 (a set of ellipsoids) is quite different from the one designed by the programmer (rectangular grids) in the previous work [9].

Table 1: Comparison with existing methods

| | Number of States | Search Time | Success Rate (%) |
|---|---|---|---|
| Previous work[9] | 243 | 500M* | 77.4 |
| Proposed method | 33 | 41M | 83.3 |
| *cf.* Fixed action length | 107 | 222M | 71.5 |

* indicates Q-learning time.

Table 1 compares the method with existing ones. Success rates are obtained from 5000 trials for each, and the number of states are counted when both the ball and the goal are observed. The search time in the previous work [9] means the learning time in terms of the period of one action primitive (33 ms). It takes about 500M (M=$10^6$) ticks because the size of the state space is much larger. The proposed method performs better than the previous work. The reductions of the size of the state space and the search time are about 1/8 and 1/12 of the previous work, respectively. For the reference, we show the result by the fixed action length of 33 ms. Compared with the previous work [9], the size of the state space and the search time are reduced into the half, but the success ratio has not been improved because the simulation has been done taking into account the delays of image processing (33 ms) and the latency of motor rotation (about 100 ms), and these effects occur when state changes, the number of which is so many due to the fixed length action. While, the size of the state space by the proposed method is small, and the size of each state is considerably larger, which is preferable for the stability of the control because the effect of sensor information processing delay becomes negligible and the stability of motion execution improves due to no changes of action commands inside one state. Only the problem due to large volume of each state is that the possibility of the incorrect merging of input vectors into wrong states seems high. This might be partly a reason why the success rate is less than 90%.

### 3.3 Experiments with a real robot

Figure 8 shows a picture of the real robot with a TV camera (Sony handy-cam TR-3) used in the real experiments. The system used in the experiments is the same one in [9]. The experiment consists of three stages. First, we collect the data obtained by real robot motions. Next, we construct the state and action spaces based on the sampled data. Finally, we control the robot based on the acquired the state and action spaces and their transition graph.

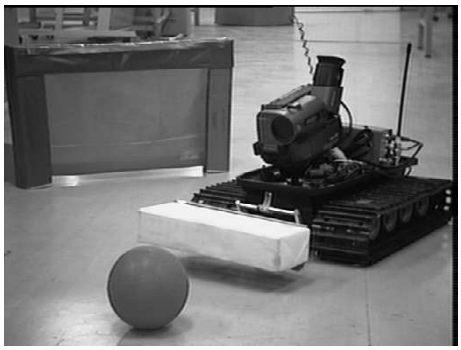


Figure 8: Our real robot

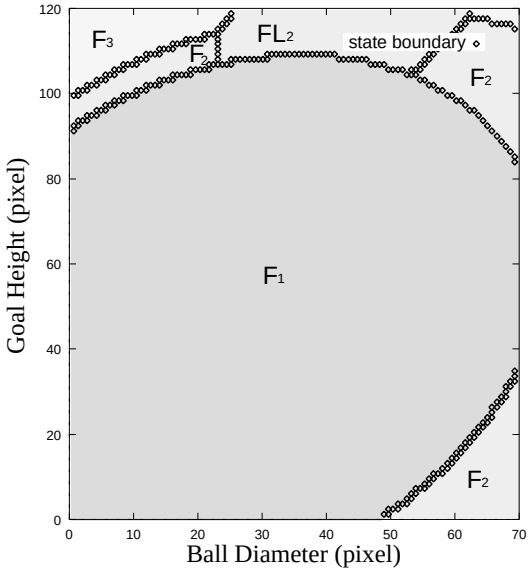



Figure 9: Result of state space construction for the real robot

The number of sampled input vectors is about 20,000 which corresponds to about 10 minutes. Figure 9 indicate the projection of the constructed state space in terms of the sizes of the ball and goal ($x_1$ and $x_3$) when the ball and the goal are observed at the center of image ($x_2 = x_4 = x_5 = 0$). Labels of regions are the same as in Figure 7, and "FL" means left forward motion. Due to the capacity of the image processor, the image size is reduced into 1/16 ($128 \times 120$), and values of each components of the input vector is also reduced into 1/4. The whole state is separated into 10 states, which is smaller than in simulation because of smaller number of experiences.

We applied the result to a real robot. Success ratio

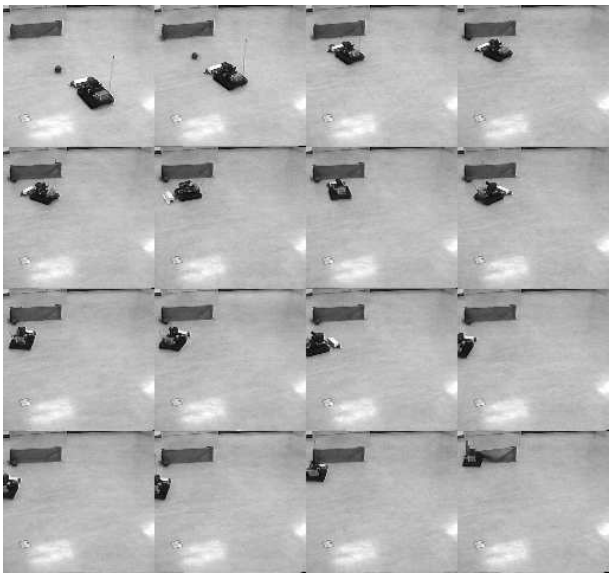


Figure 10: The robot succeeded in finding and shooting a ball into the goal

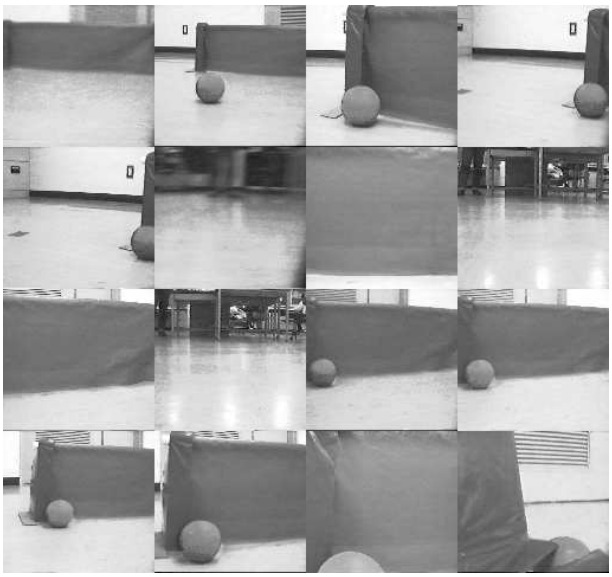


Figure 11: Images taken by the robot during the task execution

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 10 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 11 shows a sequence of images taken by the robot during the task execution shown in Figure 10. Note that the backward motion for retry is just the result of learning and not hand-coded.

### 3.4 Discussion

There are two kinds of trade offs:

- If the sampling data (input vectors) are biased, parameters of the ellipsoids change, which affects the size of the state space and the search time, and as a result a behavior also changes. In our method, we randomly initialized robot positions so that it can observe both ball and goal. However, there is no guarantee of no biases. Generally, for the less biased data, the more data and longer time are necessary. An effective method for data sampling should be developed, but there is a trade-off between the effectiveness and *a priori* knowledge on the environment and the robot.

- We used a concentration ellipsoid [12] as a model of cluster (state) of input vectors, inside which a uniform distribution is assumed. However, actual distributions are not always uniform. Ideally, situations that input vectors to be included in the model are not included and vise versa should be avoided. Complete modeling seems impossible because of uncertainties in real world. A model which includes much less error but needs complicated procedures to obtain the model parameters and/or much memory to store and recall them is not desirable because of realtime execution of robot actions. We should find a model taking into account these issues.

## 4 Concluding Remarks

We have proposed a method for constructing the state and action spaces based on experiences, and shown the validity of the method with computer simulations and real robot experiments. As described in Introduction, we regard the problem of state and action space construction as "segmentation" problem. We suppose that a physical body which can perceive and take actions in the environment is a necessary condition in order for the robot to solve this problem[4].

[4]In computer vision, "segmentation problem" has been attacked since the early stage as "image segmentation problem." Since the evaluation of the results are subject to programmers, the validity and limitation of the method seem to have been left ambiguous. From a viewpoint of robotics, segmentation of sensory data from the environment depends on the purpose (task),

The state and action spaces obtained by our method (Figure 9 indicates a projection of such a space) correspond to the subjective representation of the world for the robot to accomplish a given task. Although it seems very limited, such a representation, an inside view of the world for the robot, shows how the robot segments the world. This view is intrinsic to the robot, and based on it the robot might make a subjective decisions when facing with different environments, and further the robot might develop its view through its experiences (interactions with its environment). That is, there might be a possibility that the robot acquires the subjective criterion, and as a result, an emerged behavior can be observed as "autonomous" and/or "intelligent." To change the possibility into the real, we have to attack the following issues:

- The sensory information in our task is an image of a red ball and a blue goal filtered by color image segmentation, and their image features such as positions, areas, and orientation are used as axes of the state space. Generally, selection of features from a ordinary images is considerably a hard problem. A problem which feature is necessary to accomplish a give task might be much harder when such a feature changes depending on situations. Since use of all possible sensory information seems impossible, selection of features obtained by the given capability for feature detection is more important. For example, behavior acquisition based on the visual motion cues [13] and based on stereo disparity and motion cues [14] have been proposed. A learning mechanism for selecting features from the sensory data processing available should be developed.

- Coping with "hidden" states is another essential problem although we have not dealt with it here. This corresponds to coping with the temporal complexity of the state space structure while the above with the spatial complexity of it. How many differential operations of feature vectors are necessary and sufficient for the given task? An essential problem is selection of input vectors including the temporal axis.

- We can regard that as a result of state space construction, the action space also is temporally abstracted by defining an action as a sequence of action primitives and parameterizing its length. Since our robot has only two DOFs (degrees of freedom) spatial abstraction of action space is not necessary. However, spatial abstraction of action space is generally needed if the robot has many DOFs. For example, we human beings easily grasp something by controlling a very simple parameter (close or open your hand) although it has many DOFs physically. Both spatial and temporal abstraction of action space is necessary with state space construction.

capabilities (sensing, acting, and processing) of the robot, and its environment, and its evaluation should be done based on the robot performance.

## References


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

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

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

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

[5] D. Chapman and L. P. Kaelbling. "Input generalization in delayed reinforcement learning: An alogorithm and performance comparisons". In *Proc. of IJCAI-91*, pages 726–731, 1991.

[6] A. Dubrawski and P. Reingnier. Learning to categorize perceptual space of a mobile robot using fuzzy-art neural network. In *Proc. of IEEE/RSJ/GI International Conference on Intelligent Robots and Systems 1994 (IROS '94)*, pages 1272–1277, 1994.

[7] B.J.A. Kröse and J.W.M. Dam. Adaptive state space quantisation for reinforcement learning of collision-free navigation. In *Proc. of IEEE/RSJInternational Conference on Intelligent Robots and Systems 1992 (IROS '92)*, pages 1327–1332, 1992.

[8] H. Ishiguro, R. Sato, and T. Ishida96. Robot oriented state space construction. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems 1996 (IROS96)*, pages ???–???, 1996.

[9] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda. Vision-based reinforcement learning for purposive behavior acquisition. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, pages 146–153, 1995.

[10] H. Inoue. Research program on mechanisms for emergent machine intelligence. In G. Giralt and G. Hirzinger, editors, *Robotics Research, The Seventh International Symosium*, pages 162–170. Springer, 1996.

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

[12] H. Cramër. *Mathematical Methods of Statistics*. Princeton University Press, Princeton, NJ, 1951.

[13] T. Nakamura and M. Asada. Motion sketch: Acquisition of visual motion guided behaviors. In *Proc. of IJCAI-95*, pages 126–132, 1995.

[14] T. Nakamura and M. Asada. Stereo sketch: Stereo vision-based target reaching behavior acquisition with occlusion detection and avoidance. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, pages 1314–1319, 1996.