Cooperation of Cognitive Learning and Behavior Learning

Atsushi Ueno Hideaki Takeda Toyoaki Nishida
Department of Information Processing
Nara Institute of Science and Technology, Nara 630-0101, JAPAN
ueno@is.aist-nara.ac.jp

Abstract

Reinforcement learning is very useful for robots with little a priori knowledge in acquiring appropriate behavior. Autonomous segmentation of the continuous state space is a promising method for reinforcement learning in real robots. This is a kind of cognitive learning. We think cognitive learning should be based on similarity of rewards since the task of the robot is expressed in the rewards in the general reinforcement learning problem. Furthermore, cognitive learning should be performed in an on-line way for flexibility to changes of the environment and immediate convergence of leaning. This paper describes a learning system which can learn a state representation and a behavior policy simultaneously while executing the task. We call the system Situation Transition Network System (STNS)*. As cognitive learning, it extracts "situations" and maintains them dynamically in the continuous state space on the basis of rewards from the environment. As behavior learning, it makes an MDP model of environment and performs partial planning on the model. This is a kind of reinforcement learning. The results of computer simulations are given.*

1 Introduction

Reinforcement learning is very useful for robots with little a priori knowledge in acquiring appropriate behavior. The general reinforcement learning can be regarded as learning of a behavior policy that maximizes discounted sum of the rewards received over time. For this purpose, many reinforcement learning methods perform the approximation of the utility function (or the action-value function).

In real world, almost all robots have real-valued sensors and the continuous state space (perceptual input space). There are two approaches for approximating the utility function in the continuous state space: using an implicit representation of the function such as neural network or segmenting the continuous space into discrete states in which the value of the utility function is regarded as invariable.

As the former, several kinds of implicit representations can be used such as Perceptron-like neural network [2, 7, 10, 5, 6], CMAC (Cerebellar Model Arithmetic Computer) [8], and RBF (Radial Basis Function) [3]. But this approach has three problems: 1. it is difficult to understanding the inner representation, therefore, 2. it is very difficult to analyze the convergence and the optimality of learning, therefore, 3. the human designer must fix many parameters arbitrarily on the basis of his own experience. It is difficult for him when the task of the robot is complex.

As the latter, the most usual representation is the grid representation which the human designer must make by dividing the continuous space at equal or arbitrary intervals. It is also difficult for him in case of a complex task.

In order to cope with this difficulty, many methods have been proposed which can segment the state space autonomously. This segmentation can be regarded as a kind of cognitive learning. In this kind of learning, we think segmentation should be based on rewards. The main reason is that the task of the robot is expressed in the rewards in the general reinforcement learning problem. Furthermore, we think input vectors which are similar from the view point of rewards should be got together into the same state for a highly abstracted state representation.

Asada et al. [1] proposed a method that divides a continuous state space by hyper-ellipsoids. In this method, a state is defined 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. By this definition, it can make a highly abstracted state representation. However, it performs cognitive learning in an off-line way: the system learns a state on the basis of experiences of random behaviors in learning phase, and executes the given task on the fixed state representation. For this reason, it needs a sufficient of randam behaviors to extract a new state exactly and cannot adapt to changes of the environment.

Ishiguro et al. [4] proposed a method that first divides a continuous state space by hyper-planes on the basis of experiences of random (or given) behaviors, and then learns a behavior policy by Q-learning. It can perform these two types of learning alternately. However, it constructs fixed boundaries between states. Therefore, the states are short of flexibility.

Takahashi et al. [9] proposed a method that segments a continuous state space by the Nearest Neighbor methods while executing the task. This segmentation is based on rewards in the areas near to rewards and based on the relationship between the sensory input and its gradient in the areas far from rewards. By the latter policy, it can segment the state space stably even in the areas far from rewards. But the validity of this policy needs to be confirmed.

This paper propose a learning system which can learn a state representation (cognitive learning) and a behavior policy (behavior learning) simultaneously while executing the task. We call the system *Situation Transition Network System (STNS)*. The cognitive learning is based only on similarity of rewards in the similar way as Asadas' method: a state is defined as a set of input vectors from which the robot acquires the same reward by the same behavior. And we developed a new flexible situation representation. Therefore, our system can extract highly abstracted states. We call this highly abstracted state *situation*. The behavior learning is performed by a reinforcement learning method. In simultaneous learning of this two types of learning, good situation recognition can make a good behavior policy, and a good behavior policy can maintain well-shaped situations. This on-line learning contributes to the followings:

1. The system can adapt to changes of the environment.
2. The system can extract a situation from a few data since it can adjust its shape afterward.
3. The robot can learn the area frequently where it go frequently in task execution.
4. The robot need not wonder how long it must continue learning in learning phase.

By 2 and 3, learning converges immediately. And in compensation for the cost of maintaining situations, it can be expected that specialization to the task and flexibility to changes are realized together by interdependence between situations and behaviors.

2 Situation Transition Network System

As shown in Fig. 1, STNS consists of a situation classifier, a situation transition network (STN), and several behavior modules. In each behavior step, the system identifies the current situation where the current input is included, makes a partial plan on the STN, and activates a behavior module according to the plan. And a list of the input, the corresponding situation, the selected behavior, and the acquired reward are put into a history database. It keeps the data for a fixed period, and always has fixed numbers of data. Cognitive learning and behavior learning are performed on the data in it.

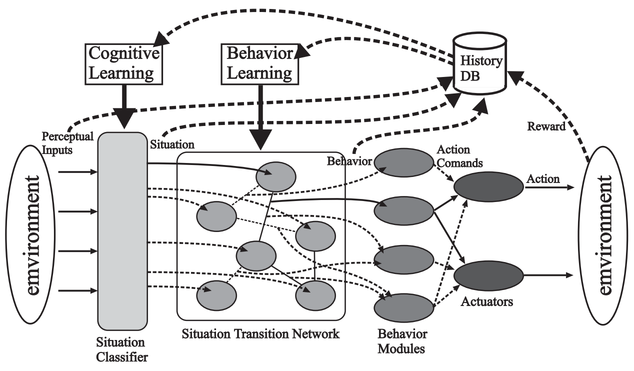


Figure 1: The structure of STNS

In cognitive learning, the system extracts situations and maintains them on the basis of rewards from the environment. This is regarded as learning of situation representation. In behavior learning, the system learns the Markov Decision Problem (MDP) model of the environment on the learned situation representation. The next behavior is decided by the partial planning on the model. This process is regarded as a kind of reinforcement learning of behavior policy. These two learning processes are performed simultaneously while executing the task.

In the next two sections, we explain the cognitive learning and the behavior learning in more detail.

3 Cognitive Learning in STNS

3.1 Segmentation Based on Similarity of Rewards

STNS segments the state space into some situations each of which has a specific meaning on the basis of similarity of rewards. The meaning of a situation is "the system can acquire the specific result by the specific behavior". The specific behavior is called the *condition behavior* of the situation. The specific results are divided into two types, i.e., *R-situation* and *T-situation*. In a situation based on immediate rewards called R-Situation, the result is to acquire a specific big reward. In a situation based on situation transitions called T-Situation, the result is to transit to a specific situation. If every chain of T-Situations is anchored to an R-Situation, every situation is guaranteed to lead to a specific reward by the same sequence of behaviors.

3.2 Bitten Hyper-Ellipsoid Representation

In on-line learning, the system should be able to decide rough shapes of situations from a limited amount of data, and to decide finer shapes as data increase. For this purpose, we propose the *bitten hyper-ellipsoid* representation. In this representation, each situation is shaped by the positive instances and the negative instances that are decided based on the meaning of the situation.

As shown in Fig. 2, this representation is a mixture of macroscopic cognition and microscopic cognition. In macroscopic cognition, the boundary of a situation is a contour of Mahalanobis' distance from the population of the positive instances. This boundary forms a hyper-ellipsoid[1] . This cognition is quick and rough that can make up for the lack of data. Microscopic cognition is realized by the Nearest Neighbor methods[2] and grows finer as data increase. By mixing these two types of cognition, a fine and flexible cognition is realized.

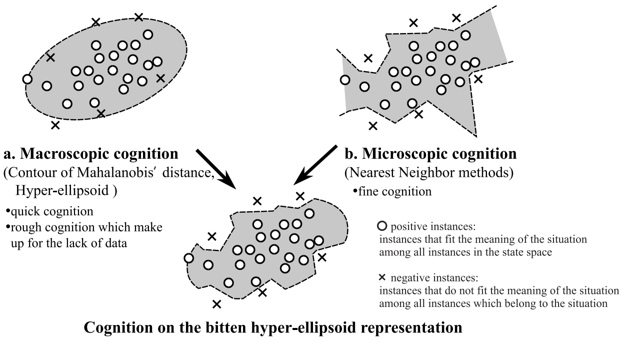

a. Macroscopic cognition
(Contour of Mahalanobis' distance, Hyper-ellipsoid)
•quick cognition
•rough cognition which make up for the lack of data
b. Microscopic cognition
(Nearest Neighbor methods)
•fine cognition
O positive instances: instances that fit the meaning of the situation among all instances in the state space
× negative instances: instances that do not fit the meaning of the situation among all instances which belong to the situation
Cognition on the bitten hyper-ellipsoid representation

Figure 2: The bitten hyper-ellipsoid representation

3.3 Cognitive Learning in STNS

In STNS, the state space is divided into overlapped bitten hyper-ellipsoids (situation 1-7) and a margin space (situation 0) as shown in Fig. 3. A new perceptual input is ascertained whether it belongs to each situation from the top of the discrimination tree in Fig. 3.

Cognitive learning is realized by extraction and maintenance of situations. There are two conditions for situation extraction.

1. **R-Situation** In the whole state space, there are enough (more than N^R_{init}) data in which the system acquired a specific reward larger than a threshold (r^R_{min}) by a specific behavior.

   T-Situation In situation 0, there are enough (more than N^T_{init}) data in which the system transited to a specific situation other than situation 0 by a specific behavior and did not acquire a large (more than r^R_{min}) reward.

2. There is no situation yet which has the specific behavior and the specific result in condition 1 as its meaning.

[1] [1] uses the same type of representation.

[2] [9] uses the same type of representation.

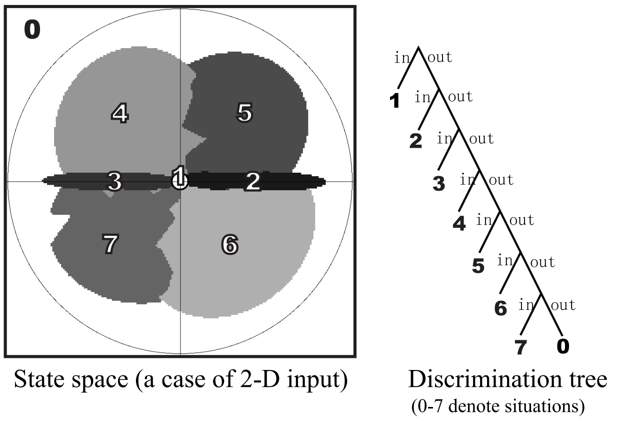


Figure 3: The state space and the discrimination tree

When both of these conditions are fulfilled, a new situation is extracted. In extraction, the meaning of the new situation is defined referring to the condition 1, and positive instances are collected from the history database.

The extracted situations are maintained by these five methods.

- Renewal of the population of the positive instances and the population of the negative instances of each situation
- Renewal of the boundary of each hyper-ellipsoid (The boundary is settled just on the farthest positive instance.)
- Renewal of the order of situations in the discrimination tree (The nearer situation to rewards is put upper.)
- Changing the meaning of deformed situations
- Removal of deformed situations

4 Behavior Learning in STNS

In STNS, each situation has a condition behavior. However, it decides a behavior policy by reinforcement learning. The main reason is that the cognitive learning in STNS is on-line learning: it extracts a situation from a few data and it is assumed that the environment can be changed, therefore the initial condition behavior is not always the optimal behavior.

In this section, we propose a new reinforcement learning method, Interleave Planning-based Reinforcement Learning (IPRL), which is suitable for STNS which has a flexible situation representation.

4.1 STN

An STN is an MDP model which consists of the transition probabilities between situations and the expectations of immediate rewards accompanying transitions. The transition probability from situation i to situation j by behavior b is called $p(i,j;b)$, and the expectation of the immediate reward which are acquired by the transition is called $r(i,j;b)$.

If Marcov property is assumed, the problem of finding the optimal policy on an STN is a Markov decision problem. Accordingly, the optimal policy is found by solving the optimal equation of dynamic programming:

$$U(i) = \max_{b \in B} \sum_{j \in X} p(i,j;b)\{r(i,j;b) + \gamma U(j)\} \qquad (1)$$

where X is the state space, B is the behavior space, i ($i \in X$) is a situation, and γ ($0 \leq \gamma < 1$) is a discounting factor. $U(i)$ is the utility of situation i, which is the expectation of the discounted sum of the rewards received over time by the optimal policy.

In STNS, the MDP model cannot be given in advance since the situation representation is changed dynamically. Therefore, the model must be estimated at the same time as the learning of a behavior policy. This problem is in the domain of reinforcement learning.

4.2 IPRL

STNS uses a new reinforcement learning method, IPRL, for behavior learning. Many reinforcement learning methods keep the estimated utilities of situations, and improve them little by little after every behavior to convergent to the optimal values. However, IPRL does not keep them but calculate them when necessary in order to adapt to the changes of situation representation immediately. For the calculation, IPRL uses a interleave planning method on STN[3] .

Interleave planning performs partial planning and plan execution alternately. As a result, it can balance between reactiveness and deliberativeness. Yamada [11] proposed interleave planning which determines the timing to switch planning into execution by the success probability of a plan. STNS uses a method similar to this, which limits the search space by the success probability. By this method, a short plan is made and executed reactively in uncertain environments, and a long plan is made and executed deliberatively in certain environments. Therefore it can balance between reactiveness and deliberativeness automatically.

A plan P in STNS is a list of pairs of a behavior and a *target situation* of the behavior:

$$P = ((b_1, d_1), (b_2, d_2), ..., (b_n, d_n))$$

where n is the length of the plan, b_i is the ith behavior in the plan, and d_i is the target situation of the behavior. The success probability of a plan is defined as the product of all transition probabilities in the plan.

IPRL calculates approximate utilities of situations using the optimal equation (Equation 1). The transition probability $p(i,j;b)$ and the expectation of the immediate reward $r(i,j;b)$ are estimated by the maximum likelihood estimation from the data in the history database.

The procedures for planning are as follows:

1. Plans are developed forward from the current situation within the limits that the success probability is larger than p_{min} and the length is shorter than n_{max}.
2. The utilities of all situations which correspond to leaves of the search tree are assumed 0.
3. The planning backtracks from leaves to the root (the current situation). In every node, the utility of the situation is calculated by Equation 1. And a pair of the behavior which maximizes the right side of the equation and situation j which maximizes the inside of $\sum$ is back-propagated to the parent node as a element of a plan.
4. When backtracking returns to the root, only one plan remains. This is the picked plan, and called the *master plan*.

Then, the system starts to execute the master plan in the environment. Plan execution is stopped and planning is started again after the last behavior in the master plan is executed, or another situation than the target situation is reached.

5 Experiments

5.1 Navigation on 2-D Input

To show the validity of STNS, we show a simple experiment on computer simulation. Fig. 4 shows the navigation problem on 2-dimensional input. Every trial starts after the goal is settled at an arbitrary position and the rover is set at an arbitrary position outside of the goal in an arbitrary direction. The trial ends when the rover arrives at the goal, and the next trial starts

[3] The policy iteration algorithm of dynamic programming is another promising method for behavior learning in STNS

immediately. At the beginning of learning, there are only the goal area and situation 0 (the margin space) in the state space. We set parameters as follows: $N^R_{init} = 3, N^T_{init} = 15, r^R_{min} = 5.0, \gamma = 0.7, p_{min} = 0.1, n_{max} = 7$.

Task:
To navigate a 16x12 rectangle rover from an arbitrary position to the only goal (a small circle whose radius is 5) in a 100x100 square continuous plane.
The goal is settled at an arbitrary position in the 72x72 square area at the center of the room.
There is no obstacle but the wall around the room.

Perceptual Input (2-dimensinal):
The real-valued (x, y) coordinates of the goal on the coordinate system fixed to the rover (the goal sensor).

Rewards:
1. Arrival at the goal: +10
2. Collision with the wall: -1
3. Trying to rotate over 90 degrees: -1
(The rover can look any direction by at most 90 degrees rotation because of the symmetry of the behaviors.)

Behaviors:
forward movement, backward movement, clockwise rotation, counterclockwise rotation.
(The rover is assumed to be able to make collision-free rotation.)

Stopping Conditions of Behaviors:
1. Getting some reward.
2. Arrival at the target situation in the master plan.

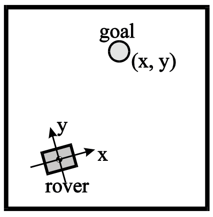


Figure 4: The navigation problem on 2-D input

Learning has converged after about 1200 behaviors are executed on average. Fig. 5 shows a typical state space after leaning has converged and the optimum state space. The small circle at the center of each space denotes states in which the rover arrives at the goal. As shown in this figure, good situation representation and good behavior policy were acquired.

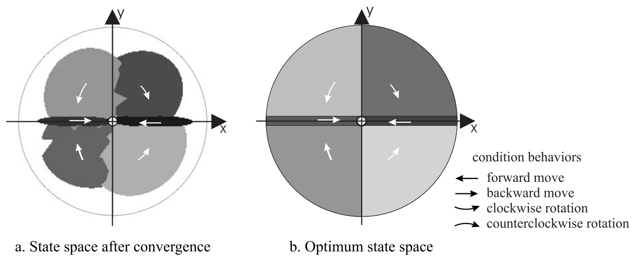


Figure 5: A state space after convergence and the optimum state space

5.2 Flexibility Test

To test the flexibility of STNS, we conducted two experiments on computer simulation. One is a flexibility test to sensor trouble and the other is a flexibility test to actuator trouble. In both experiments, we prepared an STNS after 10000 behavior steps in the above experiment (shown in Fig. 5a), changed the environment, and then let the STNS continue learning in the new environment. As changes of environment, we rotated the direction of the goal sensor in the former test, and made the revolution rate of the left wheel lower in the latter test. Fig. 6 shows a typical result in the case of 15 degrees rotation of the goal sensor and a typical result in the case of 15% lower speed of the left wheel. As shown in this figure, passably good situation representation and good behavior policy were acquired.

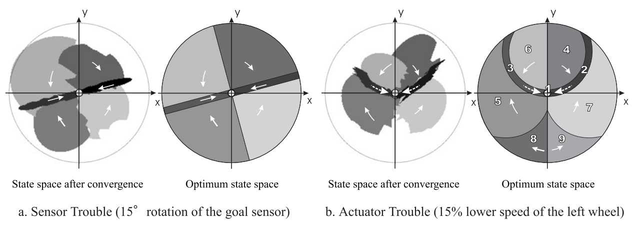

y
x
State space after convergence
Optimum state space
a. Sensor Trouble (15° rotation of the goal sensor)
b. Actuator Trouble (15% lower speed of the left wheel)
6
4
3
2
5
7
8
9

Figure 6: Results of the flexibility test

In both experiments, STNS can flexibly adapt to small changes while keeping the high performance by transforming the shapes of situations, and STNS can stably adapt to big changes by eliminating deformed and obstructive situations and starting again from the blank state space. These two types of adaptation are switched to each other autonomously. Even in the case of big changes, learning converged after a delay of at most 1000 behavior steps compared with learning from the blank state space. So it can be said that learning of STNS is stable.

5.3 Navigation on 8-D Input

The last experiment is a navigation problem on 8-dimensional input. The setting is almost the same with the previous navigation problem on 2-dimensional input. But one obstacle was settled in the room and the goal was fixed to the position shown in Fig. 7a. And the rover got a 6-dimensional obstacle sensor shown in Fig. 7b in addition to the 2-dimensional goal sensor. We set parameters as follows: $N^R_{init} = 9, N^T_{init} = 20, r^R_{min} = 5.0, \gamma = 0.7, p_{min} = 0.2, n_{max} = 7$. This problem is difficult because the dimension of the state space is high and the paths through the state space is not continuous.

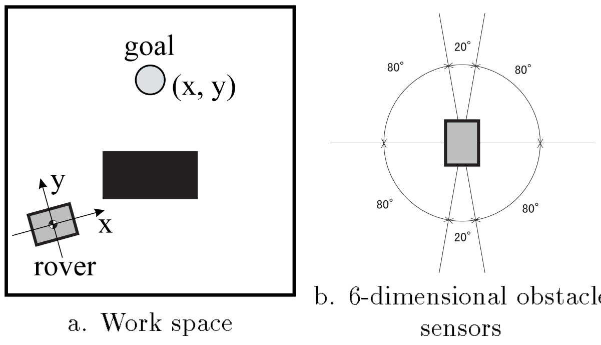
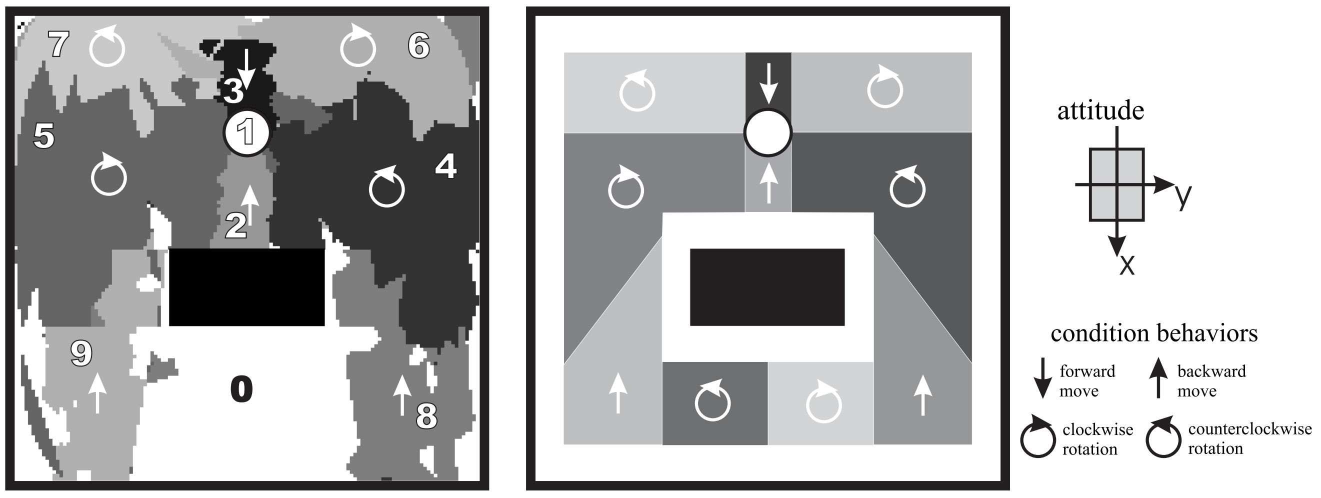


Figure 7: The navigation problem on 8-D input

Fig. 8 shows an typical segmentation after enough leaning (28000 behavior steps) and the optimum segmentation. The 8-D state space was mapped onto the 2-D work space by fixing the attitude of the rover. As shown in this figure, not bad situation classification and not bad behavior policy was obtained in the areas



a. Segmentation after enough learning b. Optimum segmentation

Figure 8: A segmentation after enough learning and the optimum segmentation

near to big rewards. However, in the areas far from big rewards, no situation was extracted. The main reason is that the ability for cognitive learning is not sufficient. For compensating for this weak point, I think hierarchical learning or empirical creation of new axes of the state space is effective.

6 Conclusion

We have proposed STNS, a method for performing both cognitive learning and behavior learning simultaneously with task execution, and shown that the method is effective to acquire good state representation and good behavior policy in continuous state space. The representation also have flexibility to changes of the environment.

This sort of dual learning system can be regarded as a symbol processing system grounded on real environments. Reinforcement learning of behavior policy is a kind of symbol processing, and learning of the state representation is regarded as learning of symbols. But reinforcement learning system can deal with symbols only for representing states or behaviors. So such systems can execute only very simple symbol processing. In order to expand this sort of symbol system, the following four functions are worth considering: structuring symbols, symbolizing objects, reusing symbols, and sharing symbols. Furthermore, in order to embed this sort of cognitive agent into the environment, parallel processing of information should be necessary.

References

[1] M. Asada, S. Noda, and K. Hosoda. Action-based sensor space categorization for robot learning. In *Proc. of IROS 96*, Vol. 3, pp. 1502–1509, 1996.

[2] A.G. Barto, R.S. Sutton, and C.W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, 1983.

[3] K. Doya. Temporal difference learning in continuous time and space. In D.S. Touretzky, M.C. Mozer, and M.E. Hasselmo, editors, *Advances in Neural Information Processing System*, Vol. 8. MIT Press, 1996.

[4] H. Ishiguro, R. Sato, and T. Ishida. Robot oriented state space construction. In *Proc. of IROS 96*, Vol. 3, pp. 1496–1501, 1996.

[5] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3/4):293–321, 1992.

[6] P. Martín and J. del R. Millán. Reinforcement learning of sensor-based reaching strategies for a two-link manipulator. In *Proc. of IROS 96*, Vol. 3, pp. 1345–1352, 1996.

[7] R.S. Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3(1):9–44, 1988.

[8] R.S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Proc. of NIPS 95*, 1995.

[9] 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 IROS 96*, Vol. 3, pp. 1518–1524, 1996.

[10] G. Tesauro. Practical issues in temporal difference learning. *Machine Learning*, 8(3/4):257–277, 1992.

[11] S Yamada. Reactive planning with uncertainty of a plan. In *Proc. of The 3rd Annual Conf. on AI, Simulation and Planning in High Autonomy Systems*, pp. 201–208, 1992.

Deadline for submission of full paper : Feb. 1, 1999