Vision-Guided Behavior Acquisition of a Mobile Robot by Multi-Layered Reinforcement Learning

Y. Takahashi **M. Asada**

Adaptive Machine Systems Graduate School of Engineering
Osaka University
Yamadaoka 2-1, Suita, Osaka 565-0871, Japan
{yasutake,asada}@er.ams.eng.osaka-u.ac.jp

Abstract

This paper proposes multi-layered reinforcement learning by which the control structure can be decomposed into smaller transportable chunks and therefore previously learned knowledge can be applied to related tasks in a newly encountered situation. The modules in the lower networks are organized as experts to move into different categories of sensor output regions and to learn lower level behaviors using motor commands. In the meantime, the modules in the higher networks are organized as experts which learn higher level behavior using lower modules. We apply the method to a simple soccer situation in the context of RoboCup, show the experimental results, and give a discussion.

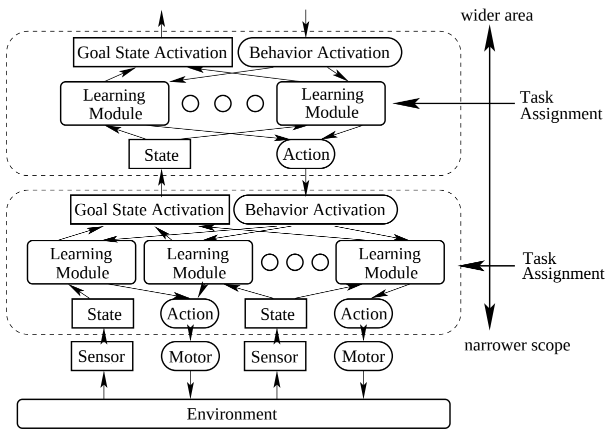
1 Introduction

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 [1]. However, its applicability to real robot tasks in a dynamic environment seems limited since the real world is too complicated for the robot to learn a variety of behaviors. Therefore, a hierarchical structure within leaning control system seems necessary, by which the control structure can be decomposed into smaller transportable chunks and previously learned knowledge can be applied to related tasks in a newly encountered situation.

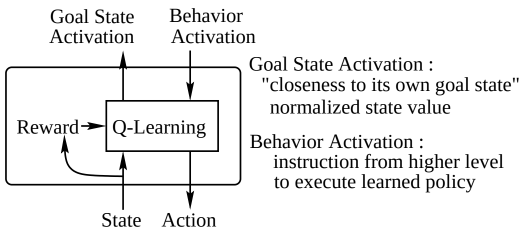
There have been a variety of works on multi-layered control architecture. Albus[2] proposed Real-time Control System, which consists of task decomposition, world modeling, and sensor processing modules at each layer of the hierarchy. The task is decomposed in advance into parallel and sequential subtasks, to be performed by cooperating sets of subordinate agents. Kaelbling[3] proposed HDG algorithm which has landmark networks as an upper level of hierarchy and enables to learn the behavior more quickly. Stone and Veloso[4] proposed layered learning, and applied it to the multi-agent learning system. At the lower-level, individual skills are learned, and more social skills are learned at the higher-level. Morimoto and Doya[5] apply a hierarchical reinforcement learning method by which an appropriate sequence of subgoals for the task is learned in the upper level, and behaviors to achieve the subgoals are acquired in the lower level. However, the problem is that the human designers have to define the subtasks, landmarks, skills, and subgoals based on their own experiences and insights.

Tani and Nolfi[6] developed an on-line multi-layered sensory flow pattern learning scheme. A set of recurrent neural net (RNN) modules is self-organized as a set of experts to account for different categories of sensory flow which the robot experiences. Meanwhile, another set of RNNs in the higher level learns the sequences of module switching observed in the lower level. Their scheme, however, doesn't have any control learning structure, which makes it difficult to acquire a purposive behavior by itself.

In this paper, we propose a method by which a hierarchical structure for behavior learning is self-organized by adding action command lines to the system in [6]. The modules in the lower networks are organized as experts to move into different categories of sensor output region and learn lower level behaviors using motor commands. In the meantime, the modules in the higher networks are organized as experts which learn higher level behavior using lower modules. Each module is assigned its own goal state by itself. We apply



(a) A whole system



(b) A behavior learning module

Figure 1: A hierarchical learning architecture

the method to a simple soccer situation in the context of RoboCup, show the experimental results.

2 Hierarchical Learning System

The robot prepares learning modules of a kind, makes a layer with the modules, and constructs a hierarchy with the layers. The hierarchy of the learning modules' layers seems to play a role of task decomposition. The lower learning modules explore small areas in the given environment, and learn lower level, fundamental behaviors. They learn behaviors with narrower scope and shorter time horizons, focusing on the more details. In contrast, the upper learning modules explore a large area, and learn higher level, more abstracted behaviors based on the learning modules at the lower layer. They have behaviors with broader scope, longer time horizons, and less concern for the details.

2.1 Architecture

The the proposed architecture of the multi-layered reinforcement learning system is shown in Fig.1, in which (a) and (b) indicate a hierarchical architecture with two levels, and individual learning module embedded in the layers.

Each module has its own goal state in its state space, and it learns the behavior to reach the goal, or maximize the sum of the discounted reward received over time, using Continuous Q-learning[8]. The state and the action are constructed using sensory outputs and motor command, respectively at the lower level.

The input and output from/to the higher level are goal state activation and behavior activation, respectively, as shown in Fig.1(b). The goal state activation g is a normalized state value [1], and $g = 1$ when the situation is the goal state. When the module receives the behavior activation b from the higher level modules, it calculates the optimal policy for its own goal, and sends action commands to the lower level. The action command is translated to actual motor command, then the robot takes the action in the world.

One basic idea is to use the goal state activations g of the lower level modules as the representation of the situation for the higher level. Intuitively, we can regard that the state value function represents how close the robot is to the goal if the module received reward only when it reach its goal, because the state value function estimate the sum of the discounted reward received over time when the robot takes optimal policy. The state of the higher level modules is constructed using the pattern of the goal state activations of the lower level modules. In contrast, the actions of the higher level modules is constructed using the behavior activations to the lower level.

2.2 Algorithm

2.2.1 Continuous Q learning

We use Continuous Q learning[8] as behavior learning module which is a modified version of normal Q-learning. We will briefly review the basics of continuous Q-learning.

First, we quantize the state action space adequately. Each quantized state and action can be the representative state and the representative action, respectively. The state and action representations are given by a contribution value vector of the representative state $(w_1^s, \cdots, w_n^s)$ and one of the representative action $(w_1^a, \cdots, w_m^a)$, respectively. A contribution value indicates the closeness to the related representative state or representative action. The summation of contribution values is one.

[1]The state value function estimates the sum of the discounted reward received over time when the robot takes optimal policy, and is obtained using Q learning.

The Q-value when executing the representative action a_j at the representative state s_i is denoted by $Q_{i,j}$. A Q-value at any state and action pair is given by:

$$Q = \sum_{i=1}^{n} \sum_{j=1}^{m} w_i^s w_j^a Q_{i,j} \qquad (1)$$

Given the representative state s_i, the optimal representative action is calculated by $\arg\max_j Q_{i,j}$. The optimal action contribution vector $\boldsymbol{a}^*$ for any state $\boldsymbol{s}$ is given by:

$$\boldsymbol{a}^* = \boldsymbol{w}^{a*} = \sum_{i=1}^{n} w_i^s e(\arg\max_j Q_{i,j}) \qquad (2)$$

where $\boldsymbol{e}(k)$ denotes an M-dimensional vector of which k-th component is one and of which others are zeros. In order to obtain the optimal action based on eq.(2), $\max Q$ is calculated by:

$$\max Q = \sum_{i=1}^{n} \sum_{j=1}^{m} w_i^s w_j^{a^*} Q_{i,j} \qquad (3)$$

Then, the Q value when choosing an action $\boldsymbol{a}$ at the current state $\boldsymbol{s}$, and transiting the next state $\boldsymbol{s}'$ given reward r is updated by:

$$Q_{i,j} \leftarrow Q_{i,j} + \alpha_t w_i^s w_j^a (r + \gamma V(\boldsymbol{s}') - Q(\boldsymbol{s}, \boldsymbol{a})) \qquad (4)$$

where $\max Q^{t'}$ denotes Q value when choosing the optimal action at the next state.

2.2.2 State Action Space Construction

Learning modules at the lowest layer construct the state action space using the sensory information and motor command of the robot. In this paper, we apply the method which is proposed in [8].

Learning modules at the higher layers construct the representative state and action using the goal state activations and behavior activations of lower modules, respectively. That is that the contribution vector of the representative state and action at the upper modules is given by the normalized pattern of the goal state activation and behavior activation of the lower modules.

2.2.3 Self-distribution of Goal State

The basic policy for the distribution of learning modules is “to assign the goal state of each learning module in the state space uniformly”. However, it seems difficult

① to find out how the state space does extend, nor

② define a distance function in the state space without robot’s experiences.

These problems will occur especially among the layers which are upper than the lowest one.

Now, we can use the state value function as the distance function which estimates the distance to its own goal state Because, we can regard that $V(s)$ represents how close the robot is to the goal if the robot received reward only when it reach its goal.

Fig.2 shows the distribution procedure of each learning module’s goal state in the state space uniformly. It shows the case of one dimension state space, however, the procedure is same way in the case of multi-dimension one. The vertical axis indicates a goal state activation of a learning module. Fig.2(a) show an example of the learning module distribution at the initial learning stage. There are three learning modules, and they are not distributed uniformly. The region where the goal state activations of other modules are low, could be judged that there is no learning modules which goal state is near. Then, the learning system adds new learning module in that place(Fig.2(b)). When the density of learning module is high, goal state activations of learning modules are high. Then, The learning system moves a learning module’s goal state to the region where the goal state activations of other learning modules will be low(Fig.2(c)). When the density of learning module is still high, the system deletes an learning modules. As a result, the goal state of each learning module distributes in the state space uniformly after the learning (Fig.2(d)). The algorithm is as follows.

(1) $S^{neighbor} = \{s^{neighbor}$ such that contribution value is larger than a threshold $T\}$

(2) If $||S^{neighbor}|| = 0$, then exit, where $|| \cdot ||$ means the number of elements.

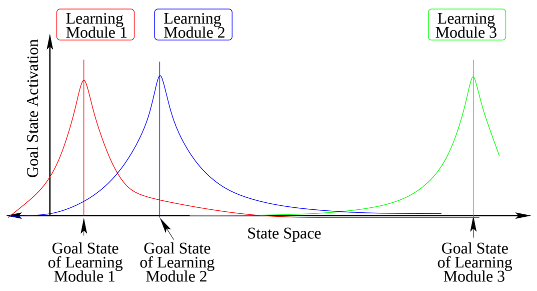
(3) Search a learning module $module_{query}$ which has its goal state in the $S^{neighbor}$

(4) Calculate the distribution of the maximum goal state activation $V_{\max}^{noquery}(s^{neighbor})$ of the learning modules which are NOT $module_{query}$

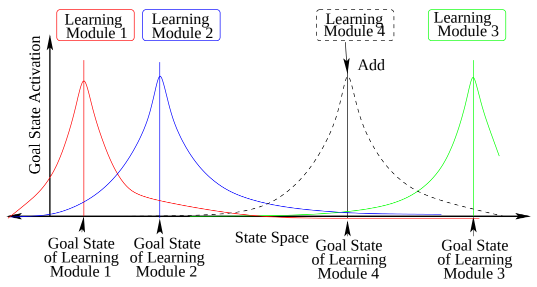
(5) If $module_{query}$ doesn’t exist, and maximum of the $V_{\max}^{noquery}(s^{neighbor})$ is low, then add a new learning modules and exit.

(6) If $module_{query}$ exists and the minimum of $V_{\max}^{noquery}(s^{neighbor}$ is high, delete $module_{query}$ and exit.

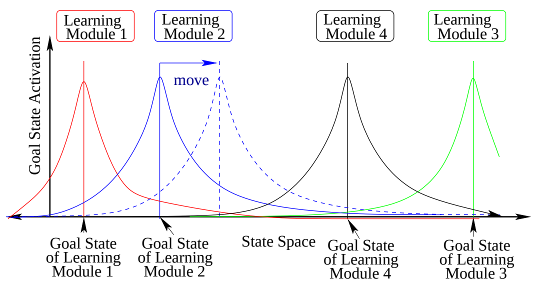
(7) Search $s_{\min}^{neighbor}$ which is the state where the $V_{\max}^{noquery}(s^{neighbor})$ is minimum among $S^{neighbor}$. If $module_{query}$ exist and its goal state is not $s_{\min}^{neighbor}$, then move the goal state to $s_{\min}^{neighbor}$



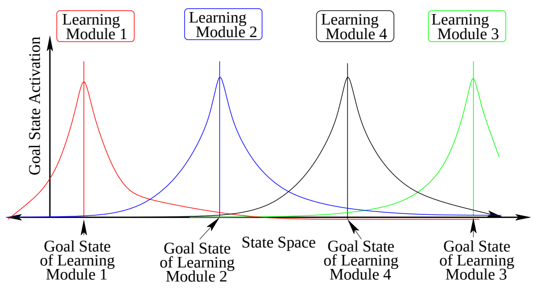
(a)Case 1



(b)Case 2

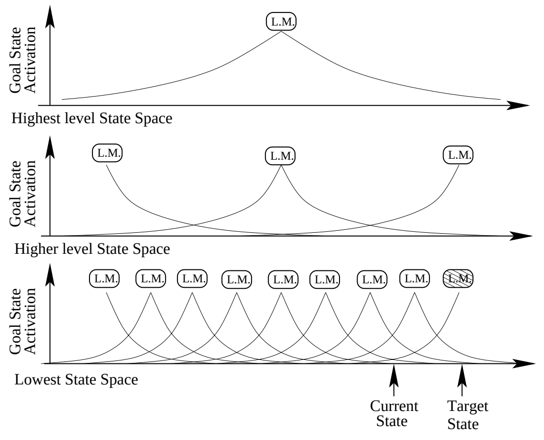


(c)Case 3

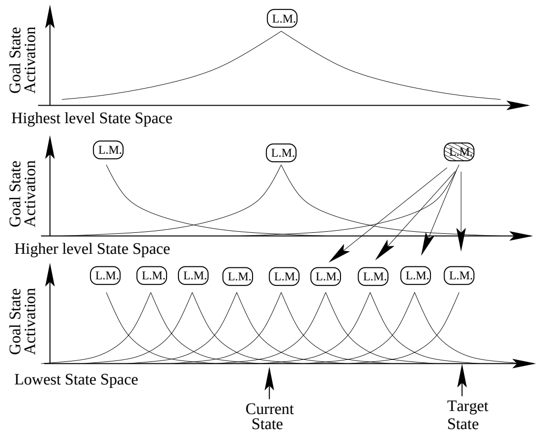


(d)Case 4

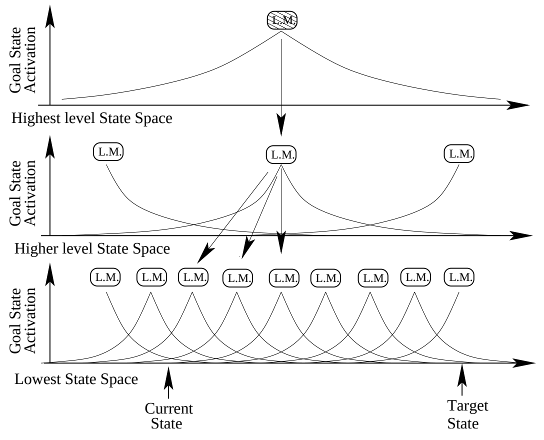
Figure 2: Example of the assignment of the goal state among learning modules



(a)Case 1



(b)Case 2



(b)Case 3

Figure 3: Strategy in the multi-layered control structure. L.M. stands for learning module

2.2.4 Construction of Layer

We've described the distribution of the learning modules among each layer. The learning system makes multi-layer by superposing them. Because it assigns less learning modules than the number of states, less number of states is assigned at upper layer. Then less learning modules will assigned at upper layer. The layering procedure stops when the number of learning module is one at the top layer.

2.2.5 Strategy in the Multi-Layered Learning System to Accomplish Task

The target state is given to the multi-layered learning system in the state space at the bottom layer. First of all, the system searches the learning module which is nearest to the target state. If the learning module can accomplish the given task, that is it can reach the target state using its policy, the system sets the behavior activation of the learning module. The system judges whether the learning module accomplish the given task or not by its Q value at the current situation. That is, if Q value is hight, then the module has a policy to reach the target state, and if Q value is low, the module has not experienced the situation, or the situation is very far from its goal state. Then, if Q value is higher than an threshold, the system judges that the module can accomplish the given task.

If the learning module $module_g^0$ which has nearest goal state to the given target state s_{target}^0at the bottom layer cannot accomplish the given task, the system lets the state at the upper layer which is related to the module $module_g^0$ be the target state s_{target}^1, and searches the learning module $module_g^1$ which is nearest to the target state s_{target}^1. If this learning module $module_g^1$ can reach the target state s_{target}^1 from the current situation, then the system sets the behavior activation of the learning module. This learning module $module_g^1$ sends its command to the lower layer by setting the behavior activations of lower learning modules, then reaches its goal state(Fig.3(b)). If the system reaches the region which the learning module $module_g^0$ at bottom layer can deal with, that is the situation becomes in case 1 in Fig.3, it starts the learning module $module_g^0$, then move to the given target state in the same way as the first step.

If the learning module $module_g^1$ cannot deal with the current situation, the system does same way at upper layer(Fig.3(c)). The multi-layered system sets behavior activation of only one learning module at each layer, because of avoidance of conflict among learning modules' policy.

3 Experiments

3.1 Setting

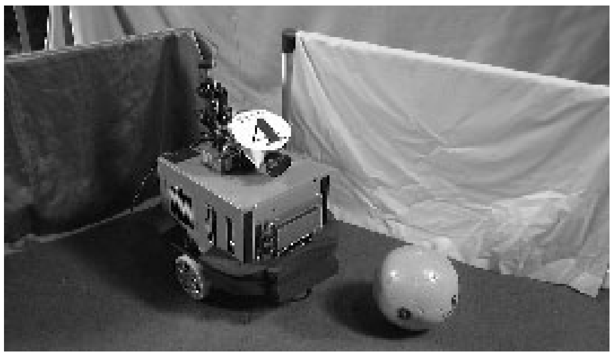


Figure 4: A mobile robot, a ball and goals

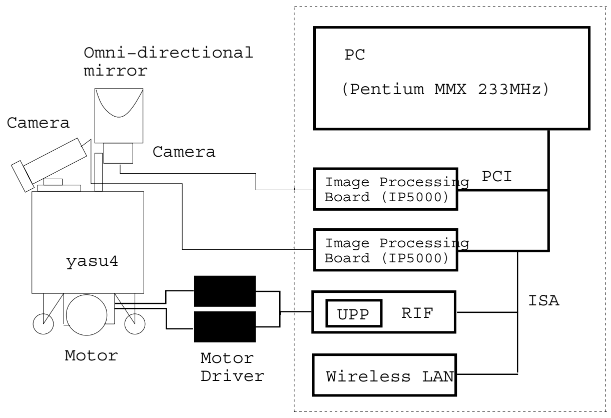


Figure 5: An overview of the robot system

To evaluate the proposed method, we apply it to a simple navigation task. The target situation is given by reading the sensor information when the robot is at the target position.

Fig.4 shows a picture of the mobile robot we designed and built, the ball, and the goal. Fig.5 shows an overview of the robot system. It has two TV cameras. One has a wide-angle lens which visual angles are 35 degrees and 30 degrees in horizontal and vertical directions, respectively. The camera is tilted down 23.5 degrees to capture the ball image as large as possible. Other has a omni-directional mirror and is mounted on the robot. The driving mechanism is PWS (Power Wheeled System), and the action space is constructed in terms of two torque values to be sent to two motors

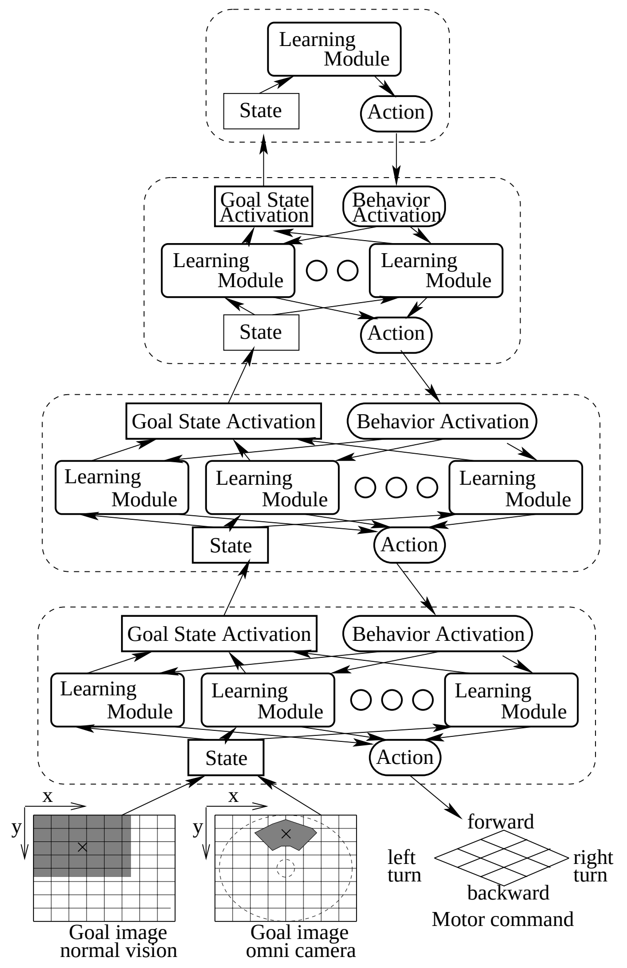


Figure 6: A hierarchy architecture of learning modules

that drive two wheels. These parameters of the system are unknown to the robot, and it tries to estimate the mapping from sensory information to appropriate motor commands by the method. The environment consists of a ball, and a goal, and the mobile robot.

In this experiment, the robot receives the information of only one goal, for the simplicity. The state space at the bottom layer is constructed in terms of the centroid of goal images of the two cameras and tessellated both into 9 by 9 grids. And the action space is constructed in terms of two torque values to be sent to two motors corresponding to two wheels and tessellated into 3 by 3 grids. Consequently, the number of representative state and action are $162 (9 \times 9 \times 2)$ and $9 (3 \times 3)$, respectively. The representative state and action at the upper layer is constructed by the learning modules at the lower layer, which are automatically assigned.

3.2 Experiment Result

The experiment is constructed with two state, one is the learning stage and other is the task execution stage using the learned result. First of all, the robot moved at random in the environment for about two hours. The system learned and constructed the four layers and one learning module exist at the top layer (Fig.6). We call each layer from the bottom, bottom, middle, upper, top layer. In this experiment, the system assigned 40 learning modules at the bottom layer, 15 modules at the middle layer and 4 modules at the upper layer. Fig.7 and 8 show the distribution of goal state activations of learning modules at the bottom layer on the state spaces of wide-angle camera image and omni-directional mirror image, respectively. The x, y axes indicate the centroid of goal images. The numbers on the figures indicate the numbers of learning modules. The figures show that each learning module is assigned on the state space uniformly.

The task for the robot is reaching a specified position using this multi-layer learning structure. The robot was located far from the goal, and faced opposite direction to it as an initial position. The target position was reaching the goal and watching it in front. Figs.9, 10, 11 show the time development of the goal state and behavior activations of learning modules at the bottom layer, middle layer and upper layer, respectively. We omitted the time development of the top learning module's activation, because only one learning module existed at the top layer, and it did never set behavior activation. The straight line segments on top of the figure indicate the development of the behavior activations. The numbers on the Fig.9 indicate the numbers of learning modules at the bottom layer, and are correspond to the same numbers on the Figs.7, 8. Fig.12 shows a rough sketch of the state transition and the commands to the lower layer on the multi-layer learning system. This figure is correspondent to the Figs.9, 10 and 11. The circles in the figure indicate the learning module, and the number in the circle indicates the number of the learning module. The up arrows indicate that the upper learning module recognizes the state which is corresponded to the lower module as the goal state. The thin solid lines indicate the state transition while the robot accomplished the task. The down arrows indicate that the upper learning module set the behavior activation of the lower learning module. When the robot located at the initial position, the learning module 25 at the bottom layer, the learning module 10 at the middle

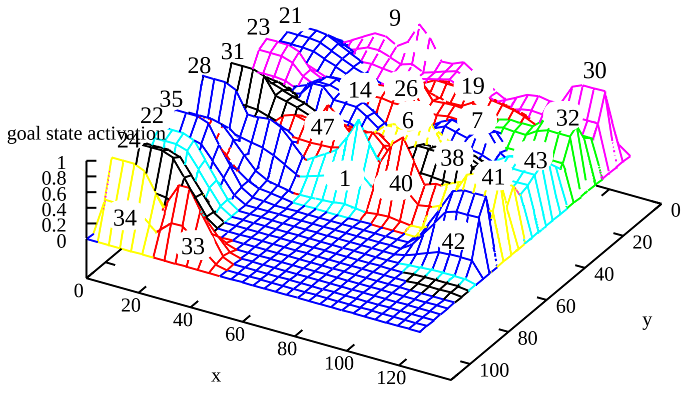


Figure 7: The distribution of learning modules at bottom layer on the normal camera image

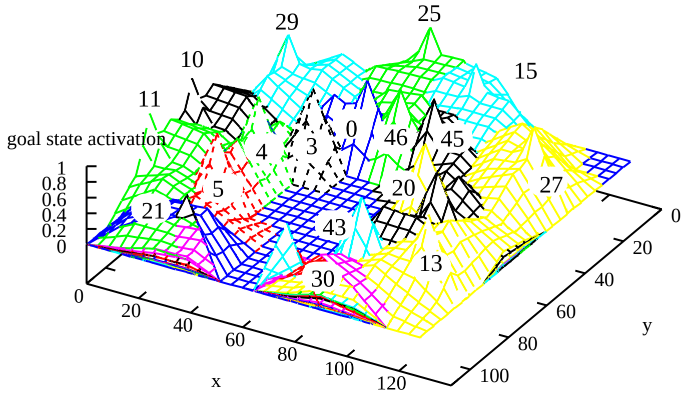


Figure 8: The distribution of learning modules at bottom layer on the omni-directional camera image

layer and the learning module 1 at the upper layer are near to their own goal states. When the robot located at the target position, the learning module 1 at the bottom layer, the learning module 7 at the middle layer and the learning module 0 at the upper layer are near to their own goal states. First of all, the system tried to activate the learning module 1 at the bottom layer, however, the module could not manage the current situation, then the system tried to activate the learning module 7 at the middle layer. But, the module could not handle the current situation, either, then the system activated the learning module 0 at the upper layer. The learning module 0 at the upper layer activated the learning module 15 at the middle layer, then this middle layer module activated the learning modules 27 and 13 at the bottom layer until about 40 step. Next, the learning module 7 at the middle layer became able to handle the situation, and activated the learning modules 30 and 26 at the bottom layer until about 360 step. Finally, the learning module 1 at the bottom layer became able to handle the situation, and the system reached the target position using this module.

4 Conclusion

This paper proposed multi-layered reinforcement learning. The results show that the competitive learning enable to determine the subgoals or subtasks by the robot without human designers intervention, and that as a whole system the robot could show purposive behaviors.

The proposed approach should be able to reuse previously learned knowledge to related tasks. And this enables the robot to learn always new, more abstracted behavior and in newly encountered situations through all its life.

Acknowledgments

This research was supported by the Japan Science and Technology Corporation, in Research for the the Core Research for the Evolutional Science and Technology Program (CREST) titled Robot Brain Project in the research area ”Creating a brain.”

References

[1] J.H. Connel and S. Mahadevan. “Introduction to robot learning”. *Robot Learning*, chapter 1, pages 1–17, Kluwer Academic Publishers, 1993.

[2] J.S. Albus. “The Engineering of Mind”. *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior (From Animals to Animats 4)*, pages 23–32, 1996.

[3] L.P. Kaelbling. “Hierarchical Learning in Stochastic Domains: Preliminary Results”. *Proceedings of the Tenth International Conference on Machine Learning*, 1993.

[4] P. Stone and M. Veloso. “Layered Approach to Learning Client Behaviors in the RoboCup Soccer Server”. *Applied Artificial Intelligence*, Volume 12, Number 2-3, 1998

[5] J. Morimoto and K. Doya. “Hierarchical reinforcement learning of low-dimensional subgoals and high-dimensional trajectories”. *The 5th International Conference on Neural Information Processing*, pages 850–853, 1998.

[6] J. Tani and S. Nolfi. “Self-Organization of Modules and Their Hierarchy in Robot Learning Problems: A Dynamical Systems Approach”. *Sony CSL Technical Report, SCSL-TR-97-008*, 1997.

[7] C.J.C.H. Watins and P.Dayan. “Technical note: Q-learning” *Machine Learning*, vol. 8, pages 279–292, 1992.

[8] Y. Takahashi, M. Takada and M. Asada. “Continuous Valued Q-learning for Vision-Guided Behavior Acquisition” *International Conference on Multisenso Fusion and Integration for Intelligent Systems* pages 716–721 1999.

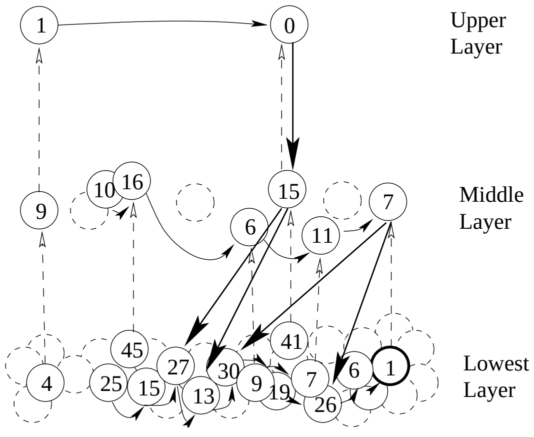


Figure 12: A rough sketch of the state transition on the multi-layer learning system

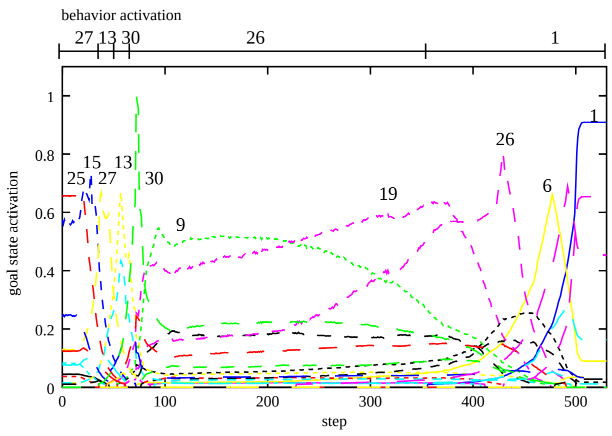


Figure 9: A sequence of the goal state activation and behavior activation of learning modules at bottom layer

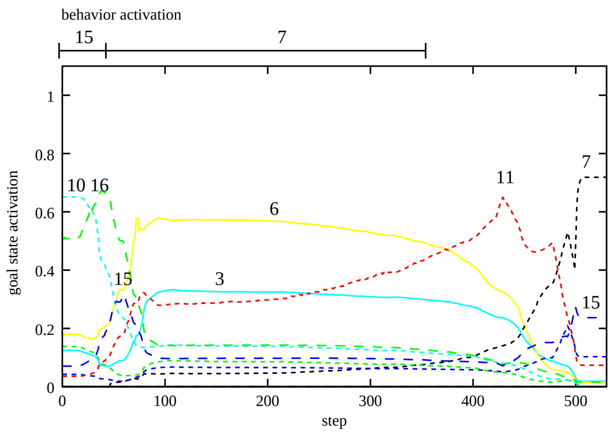


Figure 10: A sequence of the goal state activation and behavior activation of learning modules at middle layer

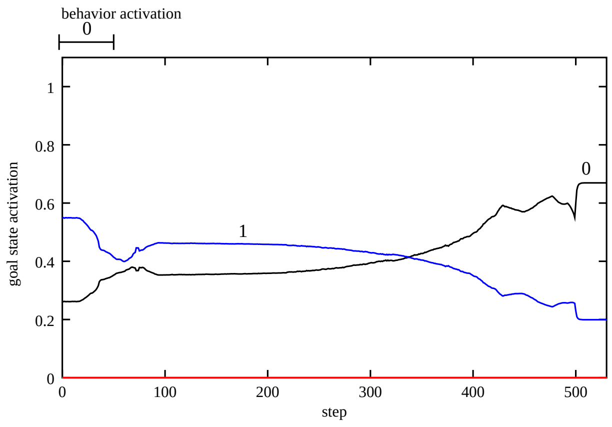


Figure 11: A sequence of the goal state activation and behavior activation of learning modules at upper layer