Incremental Purposive Behavior Acquisition based on Self-Interpretation of Instructions by Coach

Yasutake Takahashi[1,2], Koichi Hikita[1], and Minoru Asada[1,2]
[1] Dept. of Adaptive Machine Systems,
[2] Handai Frontier Research Center,
Graduate School of Engineering, Osaka University
Yamadagaoka 2-1, Suita, Osaka, 565-0871, Japan

***Abstract*— We propose a hierarchical multi-module leaning system based on self-interpretation of instructions given by a coach. The proposed method enables a robot (i) to decompose a long term task that needs various kinds of information into a sequence of short term subtasks that need much less information through its self-interpretation process for the instructions given by the coach, (ii) to select sensory information needed for each subtask, and (iii) to integrate the learned behaviors to accomplish the given long term task. We show a preliminary result from a simple soccer situation in the context of RoboCup [1].**

I. Introduction

Reinforcement learning (hereafter, RL) is an attractive method for robot behavior acquisition with little or no *a priori* knowledge and higher capability of reactive and adaptive behaviors [2]. However, single and straightforward application of RL methods to real robot tasks is difficult owing to the need for almost endless exploration that scales exponentially with the size of the state/action space, which seems almost impossible from a practical viewpoint.

Fortunately, a long time-scale behavior might often be decomposed into a sequence of simple behaviors in general, and therefore, the search space can be divided into smaller search spaces. Connell and Mahadevan [2] decomposed whole behaviors into sub-behaviors, each of which can be independently learned. However, task decomposition and behavior switching procedures are given by the designers. Morimoto and Doya [3] applied a hierarchical RL method by which an appropriate sequence of subgoals for the task is learned in the upper level while behaviors to achieve the subgoals are acquired in the lower level. In their system, the human designer has to define the subtasks based on their own experiences and insights. Doya et al. [4] have proposed MOdular Selection And Identification for Control (MOSAIC), which is a modular RL architecture for non-linear, non-stationary control tasks. However, all learning modules share the same state space. Takahashi and Asada [5], [6] proposed a multi-layered RL system. The modules in the lower networks are organized as experts to move into different categories of sensor output regions and learn lower level behaviors using motor commands. In the meantime, the modules in the higher networks are organized as experts that learn higher level behaviors using lower modules. However, this system tends to produce not only purposive behavior learning modules but also many non-purposive ones, and as a result, to require large computational resources.

When we develop a real robot that learns various behaviors in its life, it seems reasonable that a human instructs or shows some example behaviors to the robot to accelerate the learning before it starts to learn. Whitehead [7] showed that instructions given by a coach significantly improve the learning and reduce the learning time. This method, called LBW (Learning By Watching), reduces the exploration space and enables the learner to have the experience of reaching the goal frequently. Asada et al. [8] proposed a method, called LEM (Learning from Easy Missions). The basic idea is that a robot starts to learn in easy situations to accomplish a given task at the earlier stages of learning and learns in more difficult situations at the later stages to accelerate the learning the purposive behavior. They applied this idea to a monolithic learning module. To cope with more complicated tasks, this idea can be extended to a multi-module learning system. That is, the robot learns basic short term behaviors at the earlier stages and learns complicated long term behavior at the later stages based on instructions given by a coach.

In this paper, we propose a behavior acquisition method based on a hierarchical multi-module leaning system with self-interpretation of coach instructions. The proposed method enables a robot to

1) decompose a long-term task into a set of short-term subtasks,
2) select sensory information needed for the current subtask,
3) acquire a basic behavior for each subtask, and
4) integrate the learned behaviors into a sequence of behaviors to accomplish the given long-term task.

We show a preliminary result applied to a simple soccer situation in the context of RoboCup [1].

II. Basic Idea

There is a learner and a coach in a simple soccer situation (Figure 1). The coach has *a priori* knowledge

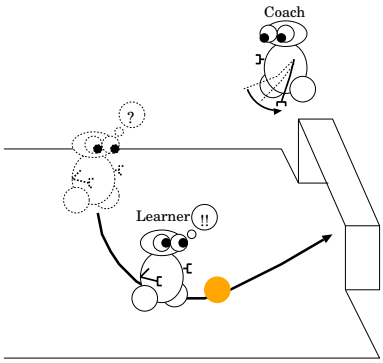


Fig. 1. Basic concept: A coach gives instructions to a learner. The learner follows the instructions and finds basic behaviors by itself.

of tasks to be played by the learner. The learner does not have any knowledge of the tasks and just follows the instructions. After some instructions, the learner segments the whole task into a sequence of subtasks, acquires a behavior for each subtask, finds the purpose of the instructed task, and acquires a sequence of behaviors to accomplish the task by itself. It is reasonable to assume that the coach will give instructions for easier tasks at the earlier stages and give ones for complicated tasks at the later stages, although it does not have any *a priori* knowledge about the learning system of the agent.

Figure 2 shows the development of the learning system through instructions given by a coach at three stages. When the coach gives new instructions, the learner reuses the learning modules for familiar subtasks, generates new learning modules for unfamiliar subtasks at the lower level and a new module for a sequence of behaviors of the whole instructed task at the upper level. After learning at one stage, the learner adds newly acquired learning modules to the learning module database. The learning system iterates this procedure from easy tasks to more complicated ones.

III. Hierarchical Multi-Module Learning System

A. Architecture

The basic idea of a multi-layered learning system is similar to [5], [6]. The details of the architecture has been extended. The robot prepares learning modules of one kind, makes a layer with these modules, and constructs a hierarchy between the layers. The hierarchy of the learning module's layers can be regarded as a role of task decomposition. Each module has a forward model (predictor) which represents the state transition model, and a behavior learner (policy planner) which estimates the state-action value function based on the forward model in an RL manner (Figure 3(b)). The state and the action

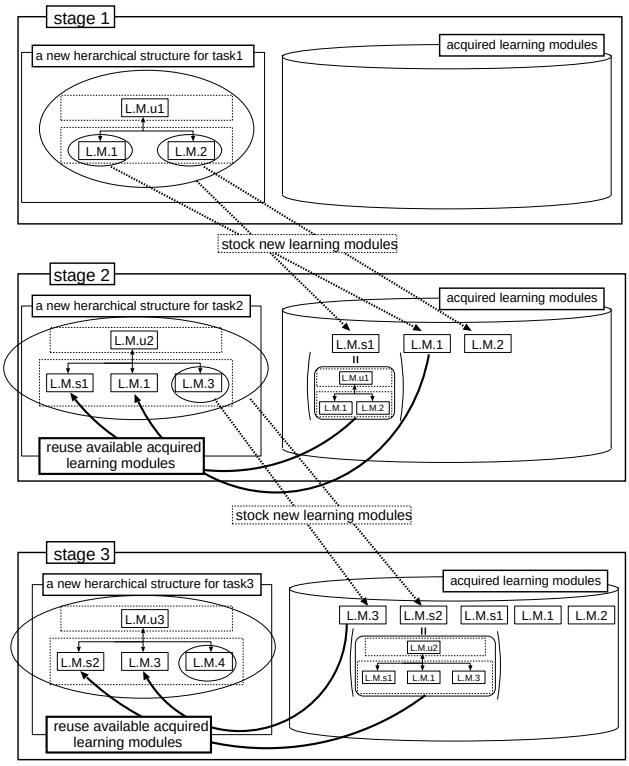


Fig. 2. The perspective of of development of the learning system with staged instructions

are constructed using sensory information and motor commands, respectively at the bottom level.

The input to and output from the higher level are the goal state activation and the behavior command, respectively, as shown in Figure 3. 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 command b from the higher modules, it calculates the optimal policy for its own goal, and sends action commands to the lower module. The action command at the bottom level is translated to an actual motor command, then the robot takes an action in the environment.

An approximated state-action value function $Q(s, a)$ for a state action pair (s, a) is given by

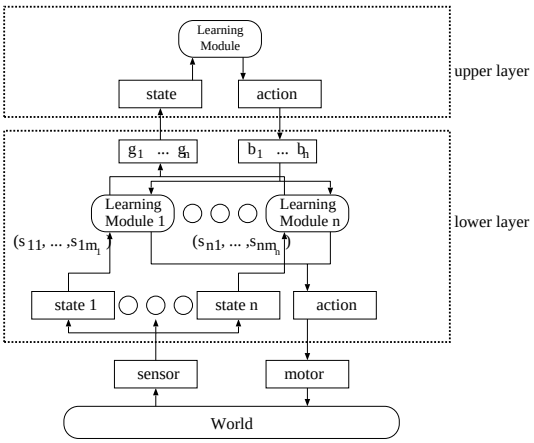
$$Q(s,a) = \sum_{s'} \hat{\mathcal{P}}^a_{ss'} \left[\hat{\mathcal{R}}^a_{ss'} + \gamma \max_{a'} Q(s', a') \right] , \qquad (1)$$

where $\hat{\mathcal{P}}^a_{ss'}$ and $\hat{\mathcal{R}}^a_{ss'}$ are the state-transition probabilities and expected rewards, respectively, and the γ is the discount rate.

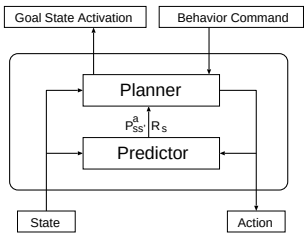
B. A Learning Procedure

The steps of the learning procedure are as follows:

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



(a) A whole system



(b) A module

Fig. 3. A multi-layered learning system

1) The coach instructs some example behaviors to accomplish a task.
2) The learner evaluates the availability of learned behaviors to accomplish the task by watching the examples.
3) The learner segments the task into subtasks, produces new learning modules at the lower layer if needed, and learns the behavior for each.
4) The learner produces a learning module at the higher layer and learns the whole behavior to accomplish the task.
5) Go to step 1.

C. Availability Evaluation

The learner needs to evaluate the availability of learned behaviors that help to accomplish the task by itself because the coach neither knows what kind of behavior the learner has already acquired directly nor shows perfect example behavior from the learner's viewpoint. The learner should evaluate a module as valid if it accomplishes the subtask even if the greedy policy seems different from the example behavior. Now, we introduce $\overline{Q}$ in order to evaluate how

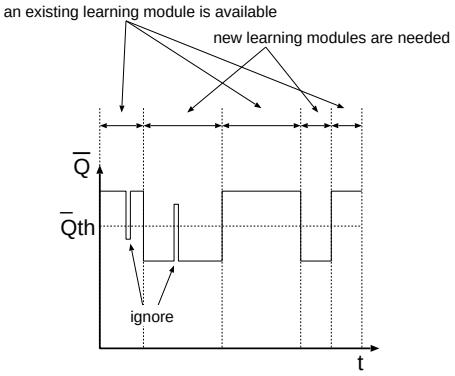


Fig. 4. Availability identification during the given sample behavior

suitable the module's policy is to the subtask:

$$\overline{Q}(s, a_e) = \frac{Q(s, a_e) - min_{a'}\, Q(s, a')}{max_{a'}\, Q(s, a') - min_{a'}\, Q(s, a')}\ , \qquad (2)$$

where a_e indicates the action taken in the instructed example behavior. $\overline{Q}$ becomes larger if a_e leads to the goal state of the module; it becomes smaller if a_e leaves the goal state. Then, we prepare a threshold $\overline{Q}_{th}$, and the learner evaluates the module as valid for a period if $\overline{Q} > \overline{Q}_{th}$. If there are modules whose $\overline{Q}$ exceeds the threshold $\overline{Q}_{th}$ simultaneously, the learner selects the module which keeps $\overline{Q} > \overline{Q}_{th}$ for longest period among the modules (see Figure 4).

D. Generating new learning modules

If there is no module which has $\overline{Q} > \overline{Q}_{th}$ for a period, the learner creates a new module which will be assigned to the subtask that has not yet been learned for the period. To assign a new module to such a subtask, the learner

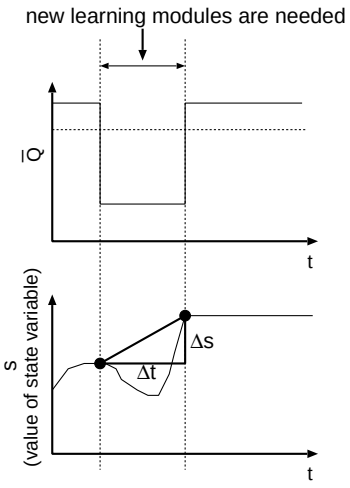


Fig. 5. Gradient of a state variable

identifies the state space and the goal state. The following shows the steps briefly.

1) Prepare a set of state spaces $\boldsymbol{S}$ and, set their priorities as $\boldsymbol{S}_i : i = 1, 2, \cdots$.
2) For each state space $\boldsymbol{S}_i$,
   a) Estimate a goal state space $\boldsymbol{G}$ in the state space $\boldsymbol{S}_i$ based on the instructed example behaviors.
   b) If the estimated goal state space $\boldsymbol{G}$ covers all of the state space $\boldsymbol{S}_i$, increment i and goto step (a).
   c) Construct a learning module and calculate Q values.
   d) Check the performance of the learned behavior for the subtask. If the success rate is low, increment i and go to step (a).
3) Add a new module based on the state space $\boldsymbol{S}_i$ and the goal state space $\boldsymbol{G}$.
4) Check the availability of modules for the given task. If there is a period where there is no available module, go to step 1.
5) Exit.

State Variables Selection: We introduce heuristics and set priorities to the set of state spaces as follows:

1) Only a few state variables are needed for all subtasks even if a large number of state variables are necessary for the whole task: We limits the number of variables to only three in this study.
2) Higher priority is assigned to a state variable that changes largely from the start to the end during the example behaviors because it can be regarded as an important variable to accomplish the subtask (see Figure 5).
3) Higher priority is assigned to the state space that has smaller average of entropy $H(s, a)$ (see equation 3) of the state transition probability $P^a_{ss'}$ for the experienced transition.

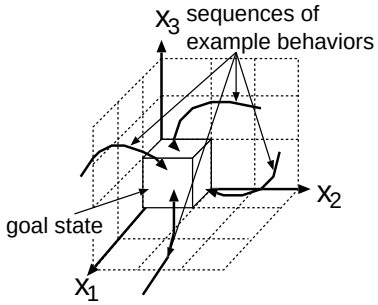
$$H(s,a) = -\sum_{s' \in S} P^a_{ss'}(s,a,s') log_2 P^a_{ss'}(s,a,s') \tag{3}$$

The reason is that the learning module acquires a more purposive behavior with a more stable state transition probability which has lower entropy [9].

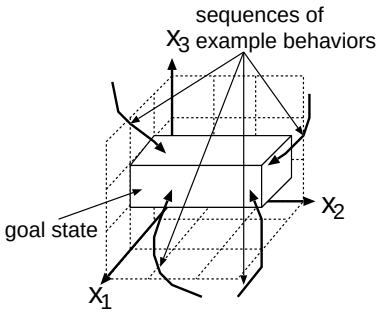
Goal State Space Selection: It is hard to specify the goal state of the subtask with a limited number of experiences of example behaviors. We need other heuristics for that.

- A state variable of the goal state tends to be the maximum, the minimum, or the medium.
- If the value of a variable has no consistency at the terminal state of the example behavior, the variable is largely independent of the goal state.

The system produces a reward model based on these heuristics.



(a) goal state is specified by all variables x_1, x_2, x_3



(b) goal state is independent from variable x_2

Fig. 6. Definition of goal state

Performance Evaluation: Even if we got a module with a proper policy based on the state transition model and reward model, there is no assurance that the module has acquired sufficient performance for the subtask. Before the system adds a new module to the available module database, it checks the success rate of the module. If the success rate is low, the system discards the module.

E. Learning behavior coordination

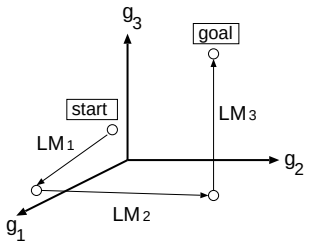


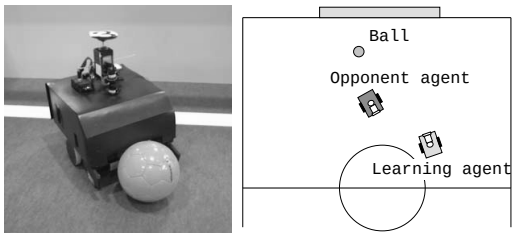
Fig. 7. An example state transition on upper layer state space

After the procedures mentioned above, there should be necessary and sufficient modules at the lower layer, for

the learning system to put a new learning module at the upper layer, and the module learns to coordinate the lower modules. The upper module has a state space constructed with the goal state activations of the lower modules. A set of actions consists of commands to the lower modules. For example there are three modules at the lower level (say LM_1, LM_2, and LM_3), then the upper module has a state space based on their goal state activations (say g_1, g_2, and g_3). Figure 7 shows an example state transition on the upper layer state space. At the initial situation, all lower modules activate low. The system sends a command to the module LM_1, then the goal state activation of LM_1, that is g_1, goes up. After LM_1 finishes its own task, the upper module sends a command to the module LM_2, and accomplishes the whole task by finally activating LM_3 at last.

IV. EXPERIMENTS

A. Setting



(a) A real robot and a ball

(b) Top view of the field

(c) Simulated camera image with a omni-directional mirror

Fig. 8. Real robot and simulation environment

Figure 8 (a) shows a mobile robot we have designed and built. The robot has an omni-directional camera system. Simple color image processing is applied to detect the ball area and the opponent in the image in real-time (every 33 msec). Figure 8 (b) shows a situation that the learning agent can encounter and Figure 8 (c) shows the simulated image of the camera with the omni-directional mirror mounted on the robot. The larger and smaller boxes indicate the opponent and the ball, respectively. The robot has a driving mechanism, a PWS (Powered Wheel Steering) system.

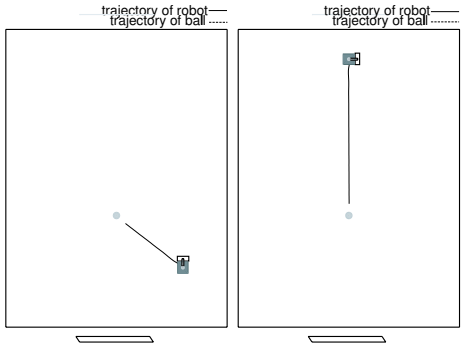


Fig. 9. Example behaviors for task 1

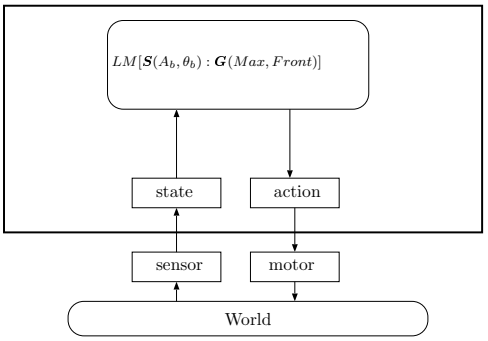


Fig. 10. Acquired hierarchical structure (task 1)

The following state variables are prepared in advance:

A_i	area of object i on the image
θ_i	angle to object i from the center of image
A_{ij}	difference of areas between objects i and j
θ_{ij}	difference of angles between objects i and j

These variables are normalized to $[0 : 1]$. The values of the area are quantized into 30 levels and the angle into 12 levels. The action space is constructed in terms of two torque values to be sent to two motors corresponding to two wheels.

B. Learning Scheduling and Experiments

The robot receives instructions for the tasks in the order given below:

Task 1: ball chasing
Task 2: shooting the ball into the goal without obstacles
Task 3: shooting the ball into the goal with an obstacle

1) Task 1: ball chasing: First, the coach gives some instructions for the ball chasing task. There is the learner, a ball, the learner's goal, and the opponent's goal. Figure 9 shows instructed behaviors for this task. According

to the learning procedure mentioned in **III**, the system produce one module $LM[\boldsymbol{S}(A_b, \theta_b) : \boldsymbol{G}(Max, Front)]$, where $\boldsymbol{S}(A_b, \theta_b)$ indicates that the state space consists of the area of ball A_b and the angle of the ball θ_b from the center of the image, and $\boldsymbol{G}(Max, Front)$ indicates that the goal state is one where A_b is the maximum value and θ_b is the front of the robot. So this module acquired the behavior of ball chasing. Figure 10 shows the constructed system for task 1.

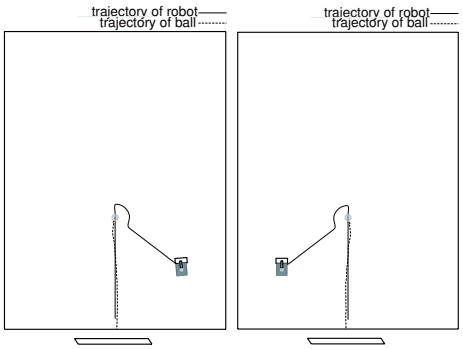


Fig. 11. Example behaviors for task 2

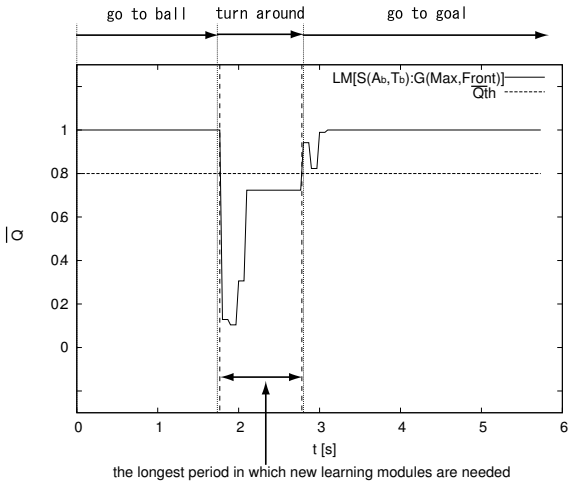


Fig. 12. Availability evaluation : before the addition (Task 2)

2) Task 2: shooting a ball into the goal without obstacles: At the second stage, the coach gives some instructions for the shooting task. Figure 11 shows example behaviors for this task, and Figures 12 and 13 show the $\overline{Q}$s for the learning modules during the example behavior performance before and after the addition of a new module, respectively. The arrows on the top of each series indicate the behavior of the instruction given by the coach. There is no valid module during the period of time from 1.7 to 2.8 seconds. The learner produces another module $LM[\boldsymbol{S}(A_b, \theta_b, \theta_{bog}) : \boldsymbol{G}(Max, Don't\ care, Min)]$ during

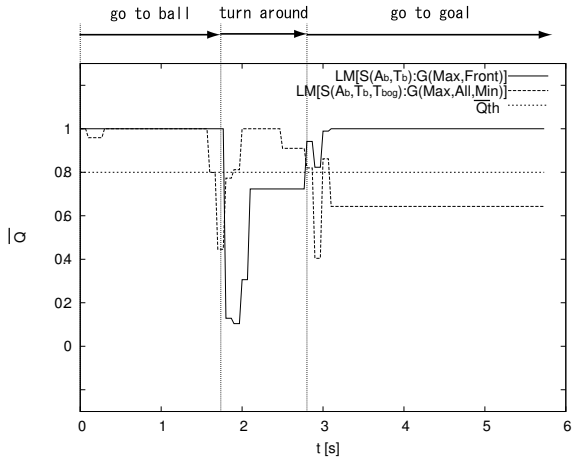


Fig. 13. Availability evaluation : after the addition (Task 2)

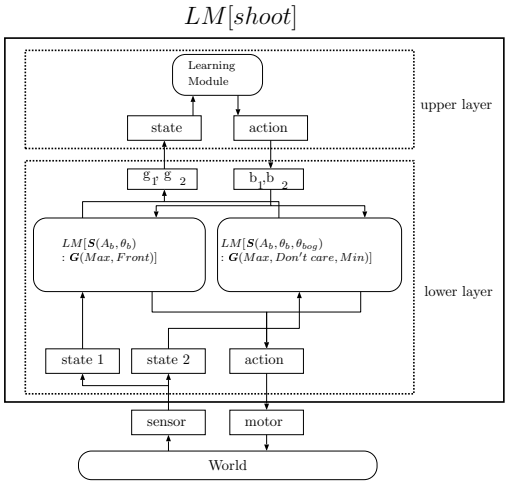


Fig. 14. Acquired hierarchical structure (task 2)

the period, where $\boldsymbol{S}(A_b, \theta_b, \theta_{bog})$ indicates that the state space consists of the area of the ball, the angle of the ball from the center of the image, and the difference between the angle of the ball and one of the goals, and $\boldsymbol{G}(Max, Don't\ care, Min)$ indicates that the goal state is one for which A_b is the maximum value, θ_b is "*Don't care,*" and θ_{bog} is the minimum value. This means that the module has a policy of going around the ball until the directions to the ball and the goal become the same. Figure 14 shows the constructed hierarchical system for this task 2. The upper module coordinates these two modules to accomplish the shooting task. Figure 15 shows the acquired behaviors for the task, and Figure 16 shows the transitions of goal state activations and the selected learning module during the behavior performance.

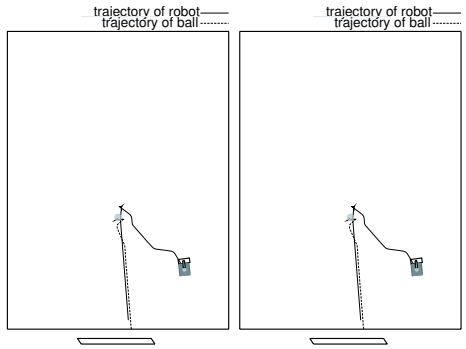


Fig. 15. Learned behaviors for task 2

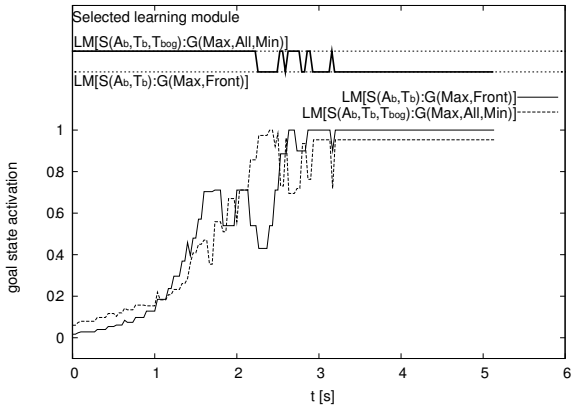


Fig. 16. Transitions of goal state activations and the selected learning module (task2)

3) Task 3: shooting a ball into the goal with an obstacle: At the last stage, the coach gives some instructions for the shooting task with obstacle avoidance. Figure 17 shows example behaviors for this task. The learner produces another module $LM[\boldsymbol{S}(A_{op}, \theta_{op}, \theta_{ogop})$:$\boldsymbol{G}(Max, Don't\ care, Max)]$, where $\boldsymbol{S}(A_{op}, \theta_{op}, \theta_{ogop})$ indicates that the state space consists of the area of opponent A_{op}, the angle of the opponent from the center of the image θ_{op}, and the difference between the angle of the opponent and the goal θ_{ogop}, and $\boldsymbol{G}(Max, Don't\ care, Max)$ indicates that the goal state is one for which A_{op} is the maximum value, θ_{op} is "*Don't care,*" and θ_{ogop} is the maximum value. Then this module acquired the behavior of going to the intersection between the opponent and the goal while avoiding collision with the obstacle. Figure 18 shows the constructed system for task 3. The upper module enables the robot to shoot a ball into the goal avoiding the opponent. Figure 19 shows the acquired behaviors for the task, and Figure 20 shows the transitions of goal state activations and the selected learning module during the task accomplishment. After learning, we applied the learned behavior to the real robot. Figure 21 shows a sequence of the experiment for the task.

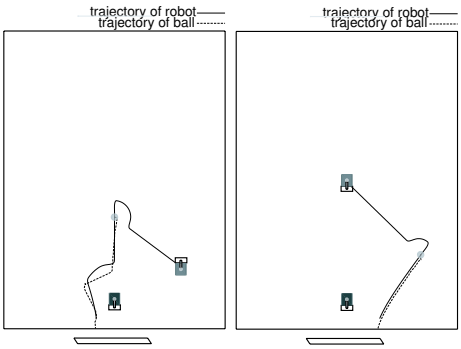


Fig. 17. Example behaviors for task 3

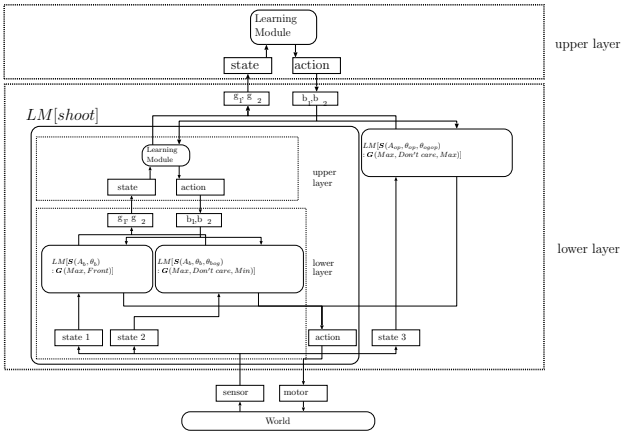


Fig. 18. Acquired hierarchical structure (task 3)

V. Conclusion

We proposed a hierarchical multi-module learning system based on self-interpretation of instructions given by a coach. We applied the proposed method to our robot and showed results for a simple soccer situation in the context of RoboCup.

VI. Acknowledgments

This research was partially 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."

VII. References

[1] M. Asada, H. Kitano, I. Noda, and M. Veloso, "Robocup: Today and tomorrow – what we have learned," *Artificial Intelligence*, pp. 193–214, 1999.

[2] J. H. Connell and S. Mahadevan, *ROBOT LEARNING*. Kluwer Academic Publishers, 1993.

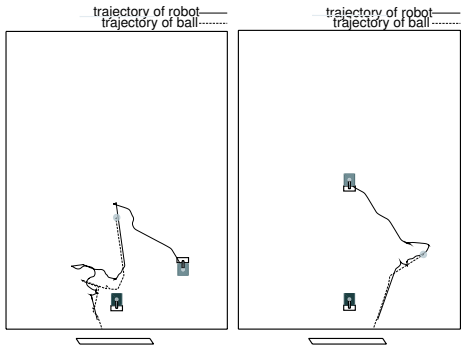


Fig. 19. Learned behaviors for task 3

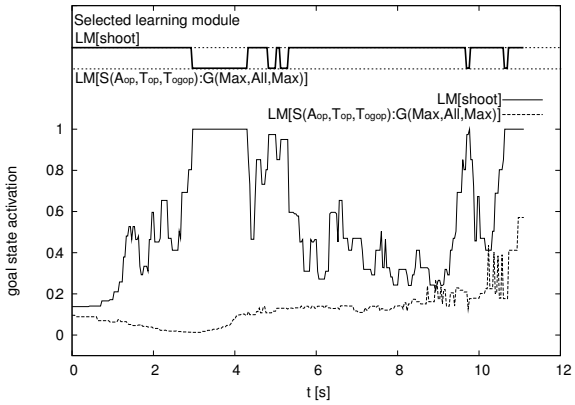


Fig. 20. Transitions of goal state activations and the selected learning module (task 3)

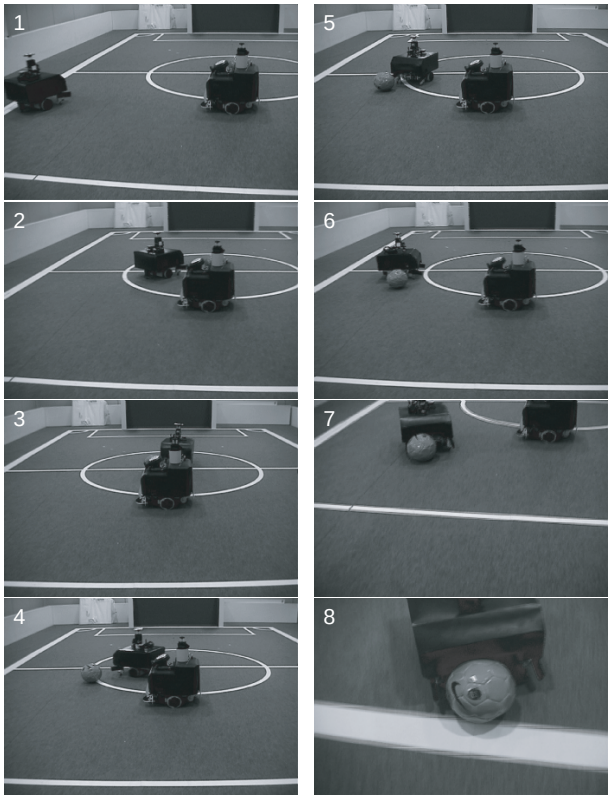


Fig. 21. A sequence of an experiment of real robots (task3)

[3] J. Morimoto and K. Doya, "Hierarchical reinforcement learning of low-dimensional subgoals and high-dimensional trajectories," in *The 5th International Conference on Neural Information Processing*, vol. 2, pp. 850–853, 1998.

[4] K. Doya, K. Samejima, K. Katagiri, and M. Kawato, "Multiple model-based reinforcement learning," tech. rep., Kawato Dynamic Brain Project Technical Report, KDB-TR-08, Japan Science and Technology Corporatio, June 2000.

[5] Y. Takahashi and M. Asada, "Vision-guided behavior acquisition of a mobile robot by multi-layered reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, pp. 395–402, 2000.

[6] Y. Takahashi and M. Asada, "Multi-controller fusion in multi-layered reinforcement learning," in *International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI2001)*, pp. 7–12, 2001.

[7] S. D. Whitehead, "Complexity and cooperation in q-learning," in *Proceedings Eighth International Workshop on Machine Learning (ML91)*, pp. 363–367, 1991.

[8] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda, "Vision-based reinforcement learning for purposive behavior acquisition," in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 146–153, 1995.

[9] T. Yairi, K. Hori, and S. Nakasuka, "Autonomous reconstruction of state space for learning of robot behavior," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)*, pp. 891–896, 2000.