A Motivation-Based Action-Selection-Mechanism Involving Reinforcement Learning

Sanghoon Lee, Il Hong Suh*, and Woo Young Kwon

Abstract: An action-selection-mechanism (ASM) has been proposed to work as a fully connected finite state machine to deal with sequential behaviors as well as to allow a state in the task program to migrate to any state in the task, in which a primitive node in association with a state and its transitional conditions can be easily inserted/deleted. Also, such a primitive node can be learned by a shortest path-finding-based reinforcement learning technique. Specifically, we define a behavioral motivation as having state-dependent value as a primitive node for action selection, and then sequentially construct a network of behavioral motivations in such a way that the value of a parent node is allowed to flow into a child node by a releasing mechanism. A vertical path in a network represents a behavioral sequence. Here, such a tree for our proposed ASM can be newly generated and/or updated whenever a new behavior sequence is learned. To show the validity of our proposed ASM, experimental results of a mobile robot performing the task of pushing-a-box-into-a-goal (PBIG) will be illustrated.

Keywords: Action-selection-mechanism, behavior-based control, reinforcement learning, robot learning.

1. INTRODUCTION

An animat-either a stimulated animal or an animal like robot-must select an action that is appropriate for the situation in which it lives in order to learn how to survive. Thus, an animat with sensors and actuators is usually equipped with an action selection mechanism (ASM) that relates its perceptions to its actions and makes it possible to adapt to its environment [1,2].

One of the most fundamental problems is deciding "what to do next" [3]. This problem is known as the action selection problem (ASP). Before deciding what to do next, the following have to be taken into consideration; (1) Is an action to be decided goal-directed? In other words, an action has to be selected in such a way that the action is the best way to reach a goal under the current situation, where nominal sequence of state-action pairs, known as a task program, can be employed as a measure of distance between the current situation and the goal situation. (2) If there are multiple goals to be achieved, what goal-directed action has to be decided among actions for different goals? However, the ASP has proven to be a hard nut to crack due to (a) incomplete knowledge, (b) unpredictable environments and surroundings, (c) imperfect sensors and actuators, and (d) limited resources [4]. As for technologies to deal with ASP, classical AI techniques [5] and several ethology-based techniques [6-11] have been proposed.

The architecture of earlier systems, which were based on traditional AI planning methods, consisted of a sense-plan-act sequential cycle and the interaction between sensing, planning, and action components. But traditional AI planning methods have some limitations, because they assume that accurate knowledge of the world's state is provided by a system's sensors. This assumption is not valid due to a number of factors, such as a changing world state, limited processing resources, and noisy unreliable sensory information [4].

On the other hand, as for alternatives of classical AI techniques, there have been proposed ethologically inspired models of action selection; insect-level and animal-level. Brooks [12] has suggested an architecture called "subsumption architecture," which is composed of competence modules with fixed priorities. This approach gives an advantage in fulfilling a goal in a complex environment. On the contrary, several researchers have proposed ethologically-inspired models of action selection [6-11,13]. Moreover, Sheutz [14] proposed a novel method to enable

Manuscript received January August 25, 2006; revised August 31, 2007 and February 26, 2008; accepted June 18, 2008. Recommended by Editorial Board member Sooyong Lee under the direction of Editor Jae-Bok Song. The work was supported by the IT R&D program of MKE/IITA (2008-F-038-01, Development of Context Adaptive Cognition Technology).

Sanghoon Lee, Il Hong Suh, and Woo Young Kwon are with the College of Information and Communications, Hanyang University, 17 Haengdang-dong, Seongdong-gu, Seoul 133-791, Korea (e-mails: shlee@incorl.hanyang.ac.kr, ihsuh@hanyang.ac.kr, wykwon@incorl.hanyang.ac.kr).
* Corresponding author.

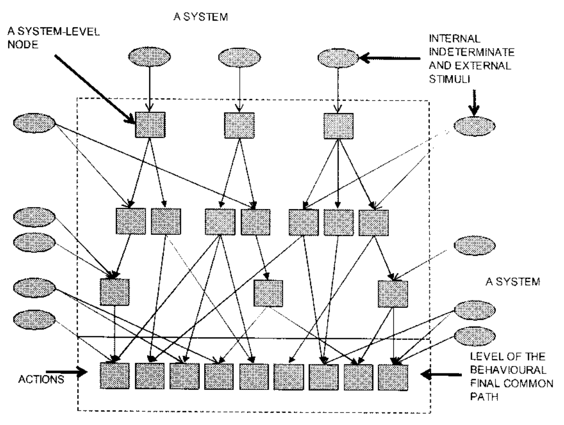

A SYSTEM
A SYSTEM-LEVEL NODE
INTERNAL INDETERMINATE AND EXTERNAL STIMULI
A SYSTEM
LEVEL OF THE BEHAVIOURAL FINAL COMMON PATH
ACTIONS

Fig. 1. Behavior selection process by motivation flows for generic ethology-based ASM[4].

dynamical switch among different action selection strategies. Those models have shown good performance in imitating real-life behavior, since action selection in those models has been based on competence modules with changing priorities. However, as shown in Fig. 1, their action has been selected not for achieving a long-term goal, but for satisfying a short-term drive generated by motivation flow. Thus, these models of action selection will not be adequate for a robot to achieve a task or equivalently a long-term goal. Furthermore, most of those works involved 'fixed' pre-designed state-action behavior systems and did not incorporate learning. Thus, they may not be appropriate in dynamic environments. Recently, several researchers have suggested ethologically-inspired models of action selection that incorporate learning [9,15-23]. However, much work remains to be done to cope with shortcomings such as lack of goal-directed action selection and lack of sequential-behavior learning. While several researches [13,23] are coping with goal-directed action selection, there is a problem with applying it to the restricted field.

In this paper, we suggest a novel architecture based on ideas from ethology that allows learning to be combined with action selection. Furthermore, we improve current ethology-based architectures to deal with sequential behaviors. Most typical tree structures organize actions into a hierarchy ranging from high-level "nodes" or activities to detailed primitive nodes via mid-level composite actions. And, in these structures, motivation flows are well described. But, behavior flows are out of focus (see also Fig. 1). Thus, only primitive actions are actually executable. Our proposed ASM, however, is designed to select the most appropriate motivation in a given situation. In addition, our ASM can allow an appropriate action to be executed for every node within that motivation. As a result, our ASM is designed to choose correct sequential behaviors to satisfy a motivation and thus enable the system to learn necessary sequential behaviors.

2. ACTION SELECTION MECHANISM

2.1. Previous works on ASM

ASMs can be generally classified as either arbitration or command fusion architectures such as [6, 24]. Arbitration mechanisms select one behavior from a group of competence modules. Arbitration mechnisms for action selection can be divided into fixed priority-based, winner-take-all, and state-based mechnisms [25].

The subsumption architecture proposed by Brooks [12] is a typical fixed-priority-based mechanism. This architecture consists of a series of behaviors, which constitutes a network of hardwired finite state machines. In the subsumption architecture, action selection is very simple: Behaviors at a higher level override behaviors at a lower level. Thus, any competence module at any given level has a priority, and high-priority modules suppress low-priority modules. The control system is hardwired directly in the structure of the behaviors and their interconnecions, and thus cannot be altered without redesigning the entire system. This type of architecture is called "fixed priority-based arbitration architecture."

Another type of arbitration architecture is the winner take-all architecture suggested by Maes [8] and Blumberg [9], which is more flexible than the subsumption architecture. In this mechanism, action selection results from the interaction of a set of distributed behaviors that compete until one behavior wins over the others. Each competence module has a priority that varies according to its own external and internal influences. Because these mechanisms are more flexible than those in a fixed priority-based architecture, learning processes can easily be incorporated.

Blumberg [9] also suggested an architecture that allows learning to be combined with action selection, based on ideas from ethology. However, that work mainly focused on "doing the right thing in a given situation." Thus, their structure selects only a single behavior to satisfy its need and learns simple state-action pairs. Note that behaviors used to achieve a mission generally consist of a series of behaviors. Selecting a single behavior in a given situation is usually not enough to accomplish a mission.

Recently, researchers had tried to adapt current ASM methods to real robot applications [26,27] or agents in virtual environment [28]. In [11], Sawada proposed EGO (Emotionally GrOunded) architecture applied to Sony QRIO SDR-4X II, where motivation flows have been employed, as in activation network of Maes. Once a motivation becomes activated by some stimuli, its corresponding action function designed by

FSM (Finite State Machine) begins to work. The situated behavior layer in EGO consists of behavior modules, where they are organized as nodes of a tree-structure, and a node becomes active as a motivation flows into the node. Here, relation of parent and child nodes of the tree is given not as a behavior hierarchy for a long term goal, but as a concept hierarchy. Thus, a sequence of behaviors from the tree will not drive the robot to the long term goal.

As we discussed earlier, sequence of behaviors has not been explicitly handled in most of the ethology based ASM. Bryson considered planning sequences of behaviors to be "reactive plans," a formalized expression which is often used in behavior-based approaches [29]. The reactive plan is a more compliated plan structure used for circumstances in which the exact ordering of steps cannot be predetermined, and consists of three elements: priorities, precondiions, and actions.

2.2. Design objectives of ASM

A robot task is usually described as a nominal sequence of state transitions from initial state s_i to goal state s_g. State transition is made by a robot behavior. However, it is noted that state can be changed without the help of any robot behaviors. For example, while a robot is approaching a person in a room to say "Hi", the person may go out of the room. In this case, the robot does not have to say "Hi", since the state that the robot is expecting has been changed to an unexpected state. That is, a state can be accidentally changed without resort to the intended behaviors of robots. It is very difficult to consider all such possible accidental state changes when programming a robot task by classical programming languages. Here, assume that all possible states can be defined for a robot and its environment. This assumption can be made to hold by defining any unexpected state as an exceptional state. Then, the problem is (1) to program a robot task as an irreducible Markov process, in which a state in a robot task program can be migrated to any state in the robot task program. This problem requires ASM to be designed as a fully connected finite state machine as shown in Fig. 2. In addition, it is necessary to know how much the current state is close to the goal state, since closeness to the goal state is a critical motivation to select actions and/or a task. (2) To fulfill such an additional requirement, a fully connected FSM as a task program needs to be described as an ordered sequence of state-action pairs. ASM has to support programming of such an ordered sequence of state-action pairs. (3) Furthermore, ASM has to be structured in such a way that some portions of such a sequence of state-action pairs for a robot task need to be learned to adapt to new situations, and/or have to be reused in another task of the robot.

To solve design problems (1), (2), and (3) as

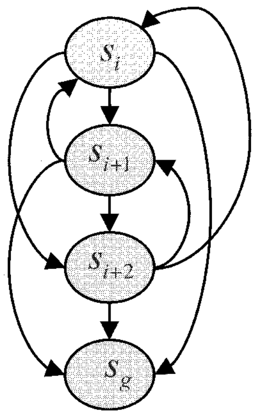


Fig. 2. An exemplar task program described as fully conneeted finite state machine.

described above, we will design Behavioral Motivaion (BM) as a basic unit to control behavior flow as well as motivation flow. For this, BM will be designed as a valued connector associated with Perception Filters (PF) and a behavior (or Action Pattern - AP), where PF is associated with promotion of motivation. PF is a function to compute degree of matching for a given context and sensory information. And, An action pattern is a sequence of primitive actions. A sequence of BM will be represented as a task program. A BM is to be designed to be dynamically inserted or deleted together with its associated PF and behavior in a task program. In this sense, such a relocatable BM will be referred to as Dynamic BM (DBM). On the other hand, a sequence of BM will be denoted as a task BM to represent a task program. To distinguish such a task BM from DBM, we will name such BM task as Static BM (SBM). Now, the order of DBM sequence is designed in such a way that the goal is located at a leaf node, and a DBM is located nearer to the goal node than the other DBMs as far as behavior associated with the DBM makes the distance to the goal shorter than behaviors in other DBMs. And, the DBM is designed such that value of a DBM flows into its child DBM, if the DBM is allowed to release its value by PF of one of its descendent DBMs. And then, a behavior is computationally selected among behaviors in a task program by comparing values of DBMs in the task program. By doing so, a sequence of DBM acts like a fully connected finite state machine. And also, nearness to the goal can be easily monitored by index of order of the DBM.

2.3. Dynamic behavioral motivation (DBM)

As explained in Sec. 2.2., a robot has to generate a series of behaviors and select the most appropriate one. To accomplish this, DBMs are organized into a flexible hierarchical network that can be changed by the learning process. A DBM has its own activation value that depends on the values from the PF, parent

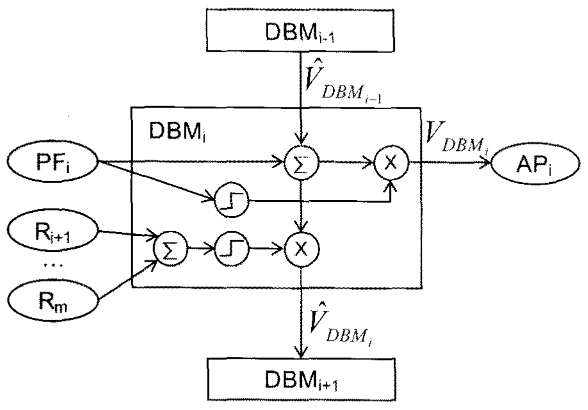


Fig. 3. A primitive node of a DBM.

node, and releasers. A releaser can be considered as a PF. But it is used to pass or block the activation value of a DBM. A DBM outputs its value to its child node while the relevant stimulus is incoming through the corresponding PF. The schematic of a DBM is illustrated in Fig. 3. The activation value is accumulated through the path while the relevant stimulus is presented by the PF. Releasers play the role of blocking the flow of activation values. The value of a DBM is given as

$$V_{DBM_i} = (\hat{V}_{DBM_{i-1}} + V_{PF_i})STEP(V_{PF_i}), \tag{1}$$

$$\hat{V}_{DBM_i} = (\hat{V}_{DBM_{i-1}} + V_{PF_i})STEP\left(\sum_{k=i+1}^{m} V_{R_k}\right), \tag{2}$$

where i, m, and k, respectively, imply the index of the ith node, the number of DBMs for a task, and the index of related releaser. And, $STEP(x)$ is defined as

$$STEP(x) = \begin{cases} 1, & \text{for } x > 0 \\ 0, & \text{for } x = 0. \end{cases} \tag{3}$$

As shown in Fig. 3, V_{DBMi} is a value where values of PFs are added into the value of the parent node's DBM. However, $\hat{V}_{DBMi}$ is the result that values of a releaser are added on V_{DBMi}. $\hat{V}_{DBMi}$ is repetitively used in the DBM calculation of child nodes. The most appropriate DBM will be selected by choosing the maximum-valued DBM, V^*_{DBMi}, which is described by

$$IndexofselectedDBM = \underset{i \in all \text{DBM for a task}}{arg \max} (V_{DBM_i}). \tag{4}$$

Now, we will show how (1)-(4) work by employing a task example as in Fig. 16(b). As shown in Fig. 16(b), nominal order of sequence for the pushing-a-box-into-a-goal (PBIG) task is given as "search→move-to-box→turn→push." After the task is initiated, it must be decided what to do next. Suppose that "PF:box and goal are not found" is active, while other PFs are inactive. Then "DBM: search" will get the largest value among DBMs for the PBIG task, which will result in activation of "AP:

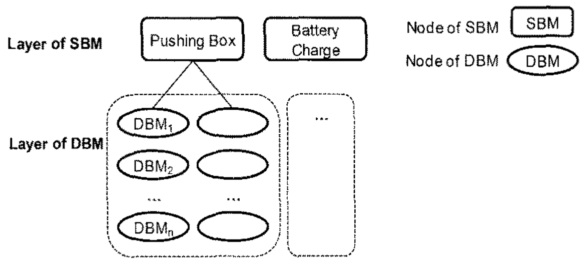


Fig. 4. Two SBMs in a flat network configuration.

search" behavior. However, if "PF: box and goal are close" is also active, this PF asks DBM (turn), DBM (move-to-box), and DBM (search) to release their values such that the values flow into DBM (push). From (1) and (2), $V_{DBM(push)} = V_{DBM(turn)} + V_{PF(push)} = V_{DBM(move-to-box)} + V_{PF(turn)} + V_{PF(push)} = V_{DBM(search)} + V_{PF(move-to-box)} + V_{PF(turn)} + V_{PF(push)}$. This implies that $V_{DBM(push)} > V_{DBM(turn)} > V_{DBM(move-to-box)} > V_{DBM(search)}$. Thus, "AP: push" behavior will be activated. By nominal order of a PBIG task, the robot is supposed to be driven to do "move-to-box" after "search" behavior. But, by an accidental transition of the environmental state from DBM (search) to DBM (push), "pushing" behavior is activated just after "search" behavior. This can be considered as an example to show equivalence between a fully connected Finite State Machine and our DBM-based ASM.

2.3. Static behavioral motivation (SBM)

An SBM implies a task. An SBM has a group of sequence DBMs to do the task. And, several SBMs (tasks) are organized as a flat network as shown in Fig. 4. Each SBM will take a value to be used for the task selection process. An SBM is activated for given external stimuli and internal needs in a flat network, when its value is the largest for SBMs in the flat network. Here, internal needs will come out from the corresponding internal state (IS) to represent drives such as hunger or thirst. The AP, which reduces a certain IS or satisfies drives, is called a consummatory action, and other APs are called appetitive actions. The value of an IS is changed after an AP is executed. The relation of an AP to an IS can be defined based on Hull's theory [30]. The value of an SBM is computed by combining values of ISs with values of PFs. In addition, an SBM receives a feedback effect from the DBM group underneath it. The value of an SBM is calculated using the equation given below.

$$V_{SBM_i} = \sum V_{IS_j} + \sum V_{PF_k} + effect^i_{DBM}, \tag{5}$$

where i, j, and k, respectively, imply the index of the i^{th} SBM, the index of a related IS, and the index of a related PF.

In (5), the term $effect^i_{DBM}$ indicates the strength showing how easily the goal can be achieved for a

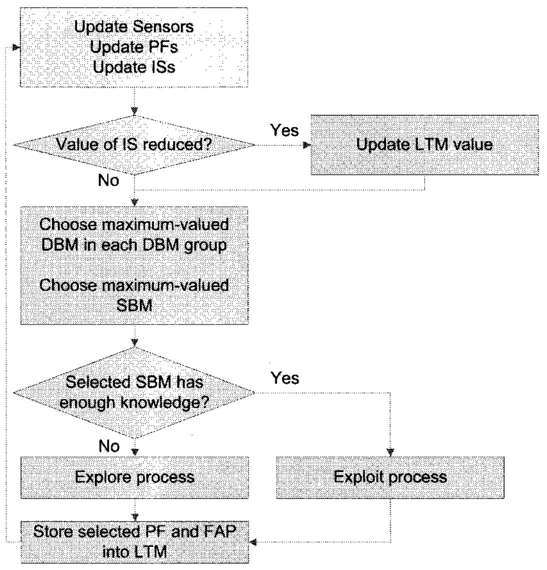

Update Sensors
Update PFs
Update ISs
Value of IS reduced?
Yes
Update LTM value
No
Choose maximum-valued DBM in each DBM group
Choose maximum-valued SBM
Selected SBM has enough knowledge?
Yes
No
Explore process
Exploit process
Store selected PF and FAP into LTM

Fig. 5. The process of our proposed action selection.

given current state of the environment. Thus, the value of an SBM may be high not only when the needs of the SBM become more important than those of other SBMs, but also when its goal is believed to be easily achieved in the current state of the environment.

Fig. 5 shows a summary of process of our proposed action selection mechanism integrating learning processes. After the SBM of the largest value is chosen, one of the following two processes is activated in order to select an action: ‘exploit’ and ‘explore’. An ‘exploit’ process is performed when a BM system has enough knowledge to satisfy its motivation. The ‘exploit’ process is performed by executing the most appropriate AP (or behavior) for a given situation. On the other hand, the ‘explore’ process is activated whenever a DBM or a sequence of DBMs is not available for an SBM. This process always requires us to have an effective learning system integrated with the ASM.

3. SHORTEST PATH-BASED REINFORCEMENT LEARNING OF DBM

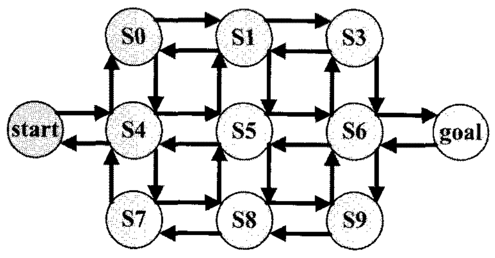
As described in Section 2, a task of a robot can be effectively programmed in the form of a fully connected finite state machine by employing our proposed DBM and SBM. However, it is still necessary to carry out numerous programming trials as well as to use sufficient domain-specific task knowledge. Environmental states from the viewpoint of robot sensors will most often be different from those from the viewpoint of task programmers, primarily due to lack of precise perception using contemporary sensors with reasonable price. This difference may cause a robot to meet unknown situations. Thus, to cope with such a programming difficulty, a robot has to be endowed with the capability to learn a sequence of ordered DBM for a task or a subtask.

In this section, a model-based reinforcement learning technique [31] is used, where the model will be progressively refined whenever a successful sequence of behaviors is achieved. Specifically, under a current model, shortest paths are found from all states to the goal, where Q-value is increased primarily for the state around the shortest-path. And, exploitation and exploration are performed based on Q-values of the states. If there is found another successful sequence of state-behaviors, then it will be used to refine the model again. And the same process as above is repeated.

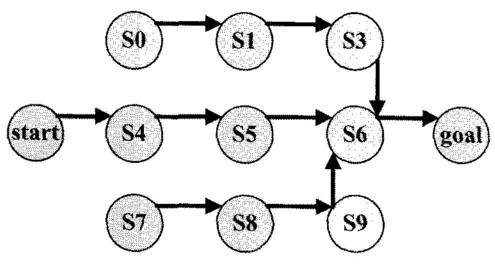
Fig. 6 presents our proposed Shortest Path-based Reinforcement Learning (SPRL) process. In Fig. 6, a Long Term Memory (LTM) consists of s, a, s' and v_{LTM}, where s is a current state, a is a current action, s' is the next state which is obtained by action a in a state of s, and v_{LTM} is a reliability value. LTM is updated whenever a learning episode is completed. An episode is a trajectory of state-action pairs for a task which starts from an initial states, and is terminated at a goal state, or a state violating time bound. Note that all states and actions of the LTM can be represented as nodes and edges of a direct acyclic graph as shown in Fig. 7(a). Thus, an LTM may include several paths from all states to a goal state. Among several paths, as shown in Fig. 7(b), we can find shortest-paths from all states to a goal state in the graph model. Now, Q-values of state-action pairs around shortest paths are increased, and Q-values of other state-action pairs will not be increased. Thus, it is expected that state-action pairs around shortest paths are selected with a relatively high probability. This type of Q-value update strategy can be considered as a process to reduce exploration space. And thus, it is expected that the number of learning episodes is drastically lowered

```
INITIALIZE $Q(s,a)$ and $LTM(s,a,s')$;
LOAD learning paramters $\alpha$, $\gamma$, $\eta$;
DO forever
  choose $a$ from $s$ with respect to $Q$ value;
  execute $a$;
  observe $s'$ and $r$;
  $Q(s,a) = Q(s,a) + \alpha[\gamma \max_{a'} Q(s',a') - Q(s,a)]$;
  update $LTM(s,a,s')$;
  IF $s'$=goal state THEN
   goal state $g = s'$;
   $findShortestPath(g)$ $\leftarrow$ shortest paths from
                    all state to $g$;
   REPEAT for all $s$, $a$
     IF ($s$, $a$) in shortest path THEN
       $Q(s,a) = \eta(1 - Q(s,a)) + Q(s,a)$;
       $v_{LTM(s,a,s')} = \eta(1 - v_{LTM(s,a,s')}) + v_{LTM(s,a,s')}$;
```

Fig. 6. Shortest path-based reinforcement learning (SPRL) process.



(a) all paths.



(b) shortest paths.

Fig. 7. An example of shortest path finding.

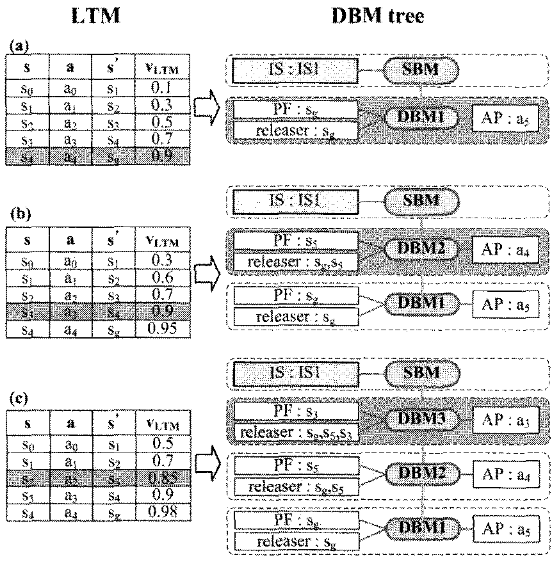


Fig. 8. An example of inserting DBMs form LTM.

by combining Q-learning with a shortest path finding method.

A learned tuple of LTM, (s, a, s') with values that exceed a certain threshold will be added to the BM tree. Fig. 8 shows the process where LTM entries having the high reliability are added to the DBM tree. Next, a process in which learned state-action pairs are added to the DBM tree will be illustrated. Note that Fig. 8 shows the reliability values of LTM after some trials were performed, and v_{LTM} of (s_4, a_4, s_g) exceeds the threshold. In this paper, we set the threshold value to 0.8. Because a tuple, (s_4, a_4, s_g) of the LTM has not been included in the corresponding DBM tree, this tuple will be added to the branch of the SBM. After more trials are performed, the LTM may be changed as shown in Fig. 8(b). In Fig. 8(b), there are two tuples with values exceeding the threshold. Because the value of (s_3, a_3, s_4) exceeds the threshold, the tuple will be added to the DBM tree. The position in which the new tuple is added is a parent node of a certain DBM. A state of PF in that DBM is the same as s' in the tuple. To reach the goal, the action of a_3 in (s_3, a_3, s_4) must precede the action a_4 in (s_4,a_4,s_g). Thus, the position of that entry will be the parent node (s_4, a_4, s_g). After many trials, a new appetitive behavior to reach the goal can be added to the BM tree as shown in Fig. 8(c).

4. COMPARISONS WITH OTHER ASMS

When the proposed ASM method is compared with other existing behavior-based AI methods or the ethology-inspired ASM methods, the following

Table 1. Comparisions of capacity of ASMs.

Capacity / ASM	structure of behavior	behavior sequencing	behavior coordition strategy	incoporation of learning
proposed ASM	hierarchical	O	dynamic priority	O
subsumption [12,32]	flat	X	fixed priority	X
activation network [8]	flat(network)	O	dynamic priority	X
reactive plan [29]	flat	O	dynamic priority	X
ethology-based [9,11,33]	flat	X	fixed priority	X
hierachical FSM-based [13,34]	hierarchical	X	fixed priority	X
RL-based [16,20,22]	flat	X	dynamic priority	O

characteristics are observed: (1) hierarchical representation of behaviors, (2) goal directed sequential behavior generation, (3) dynamic priority based behavior coordination strategy, and (4) incorporation of learning.

To better clarify the characteristic of the proposed ASM method, there will be shown a comparison of our proposed ASM with the well-known ASM methods in Table 1. In the first column of Table 1, 'structure of behavior' shows whether behaviors of the ASM are hierarchically comprised or not. If behaviors are hierarchically comprised, robot programming can be more instinctive. Moreover, a group of behaviors which consists of robot programming can be reused. In the second column of Table 1, 'behavior sequencing' shows whether ASM can produce and manage the sequence of behavior or not. 'Behavior sequencing' means whether the previously chosen behavior has an effect on the selection of behavior. In the third column of Table 1, if 'behavior coordination strategy' is fixed, the priority of a behavior cannot be changed. Therefore, in the same external condition, a fixed priority-based ASM output only has the same behavior sequencing. However, the dynamic priority-based behavior coordination strategy helps an agent to show various kinds of behavior sequencing. In the fourth column of Table 1, 'incorporation of learning' means that action selection strategy can be modified by a learning mechanism. ASM with 'incorporation of learning' helps a robot to accomplish a task in a dynamically changing environment.

5. EXPERIMENTAL RESULTS

5.1. Experimental mobile robot system

Our own mobile robot is designed and employed for our experiments as shown in Fig. 9. The robot has been designed to have a single CCD camera with pan/tilt guide, and to be controlled by a two-wheeled differential drive system and an embedded PC (VIA C3).

5.2. Experimental objectives

To show the validity of our proposed ASM integrating reinforcement learning, experimental set-

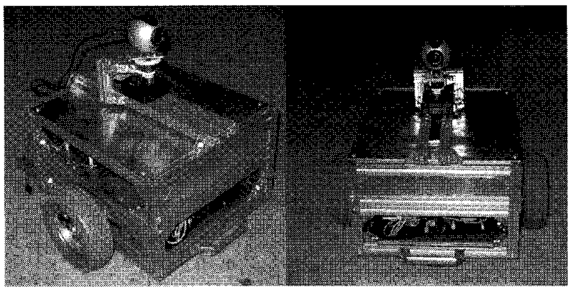


Fig. 9. Photos of our designde mobile robot.

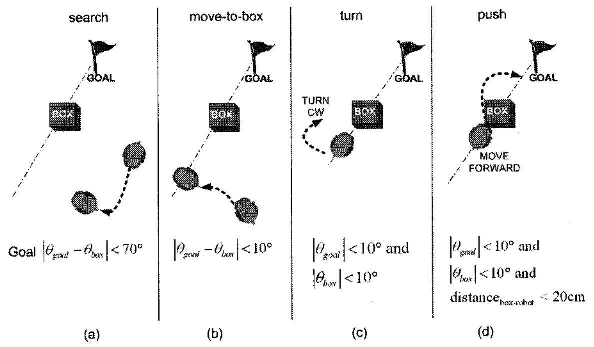

search
move-to-box
turn
push
GOAL
GOAL
GOAL
GOAL
BOX
BOX
BOX
BOX
TURN CW
MOVE FORWARD
Goal $|\theta_{goal} - \theta_{box}| < 70°$
$|\theta_{goal} - \theta_{box}| < 10°$
$|\theta_{goal}| < 10°$ and $|\theta_{box}| < 10°$
$|\theta_{goal}| < 10°$ and $|\theta_{box}| < 10°$ and $\text{distance}_{box\text{-}robot} < 20\text{cm}$
(a)
(b)
(c)
(d)

Fig. 10. A task description for PBIG task.

up for a pushing-a-box-into-a-goal (PBIG) task of a mobile robot is organized as shown in Fig. 10.

In this experiment, it is shown that a mobile robot can learn action patterns to push a box into a goal by our SPRL. For this, a robot is made to receive a reward whenever it completes the PBIG task successfully.

5.3. Implementation notes

It is a first note of object extraction from a CCD Camera Image. An image of an object from a camera needs to be separated from the background to recognize the object. For this, in the phase of color image processing, HSV (Hue, Saturation, Value) representation is employed to minimize the variational effect of light intensity. After getting the images of objects separated from the background image, a center of gravity is computed. And then, the center of gravity is mapped into a relative angle of the object and the area is mapped into the relative direction of the object.

It is a second note of Bayesian filter-based design of logical sensors. For a pushing-a-box-into-a-goal task, a robot must recognize locations of objects such as boxes and a goal. It is recalled that, our robot has only a single camera to find objects. And, due to the limited field of vision of the camera, our robot could not recognize locations of all objects around the robot at any time. Therefore, our robot is required to have a logical sensor to process perceptual state information, especially to estimate locations of out-of-sight-objects. To estimate locations of multiple objects, Bayesian

$state := < \alpha, \beta, \gamma, \delta >$
$\alpha = 0$ when $|\theta_{box} - \theta_{goal}| < threshold$
$\alpha = 1$ when a box is left of a goal
$\alpha = 2$ when a box is right of a goal

$\beta = 0$ when $|\theta_{box} - \theta_{goal}| < 10^o$
$\beta = 1$ when $|\theta_{box} - \theta_{goal}| < 70^o$
$\beta = 2$ when $|\theta_{box} - \theta_{goal}| < 120^o$

γ: state-division according to the angle of a box
δ: state-division according to the angle of a box

Fig. 11. Definition of states.

Table 2. Definition of primitive actions.

Name	forward movement (mm)	Heading angle movement (degree)
TURN_L	0	5
TURN_R	0	-5
MOVE_FL	20	5
MOVE_FFL	40	5
MOVE_F	40	0
MOVE_FFR	40	-5
MOVE_FR	20	-5
BACK	-40	0
BACK_L	-20	5
BACK_R	-20	-5

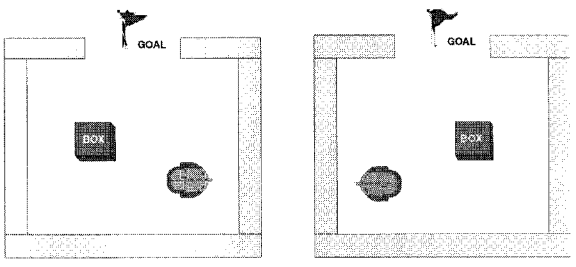
filter (grid-based) is used here. The key idea of Bayesian filter is to estimate the posterior probability density over the state-space condition on the data [35]. It is a third note of state organization and primitive actions for action patterns. Action patterns are sequences of primitive actions to satisfy specified subgoals. The relative location between the robot and objects such as a box and a goal are used as perceptual states for the action pattern. State organization and primitive actions for action patterns for a PBIG task are shown in Fig. 11 and Table 2.

5.4. Learning of action patterns

Four action patterns are necessary to accomplish the PBIG task as shown in Fig. 10. Among them, three action patterns of "move-to-box", "turn", and "push" are learned from the scratch by SPRL. To learn an action pattern, 10 trials are performed at two different initial locations as shown in Fig. 12. Because a robot is assumed to have no initial knowledge on the action patterns, the robot often gets away in a 2m x 2m experimental playground. Such overruns from the playground are regarded as a failure, and trajectories of failed trials are excluded in learning action patterns. The learning parameters of the SPRL process in Fig. 6 are given for the experiment as follows; $\alpha = 0.8$, $\gamma = 0.8$, $\eta = 0.2$. Performance of the action pattern by SPRL is shown in Fig. 13 to be compared with those by Q-learning and hand-coded action patterns. It is observed from Fig. 13 that performance of the action pattern by SPRL is almost similar to those by the other two action patterns, and convergence of SPRL is faster than that of Q-learning. To compare performance of three different methods in terms of the number of steps for the whole PBIG task, the averaged number of primitive actions for each action pattern after 10 trials of learning are shown in Table 3 for the methods of hand-coded, SPRL, and Q-learning, respectively. It is observed from Table 3 that the action pattern for the hand-coded method requires 55 steps, and the action patterns for SPRL, and Q-learning require 65 steps and 70 steps, respectively for the whole PBIG task. From this experiment, the action patterns learned by SPRL method are observed to

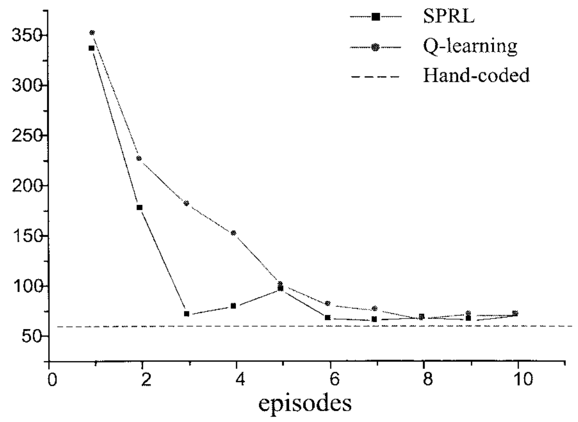
Table 3. Averaged number of steps in action patterns.

	hand-coded	SPRL	Q-learning
move-to-box	24	32	31
turn	20	20	24
push	11	17	18
total	55	65	70

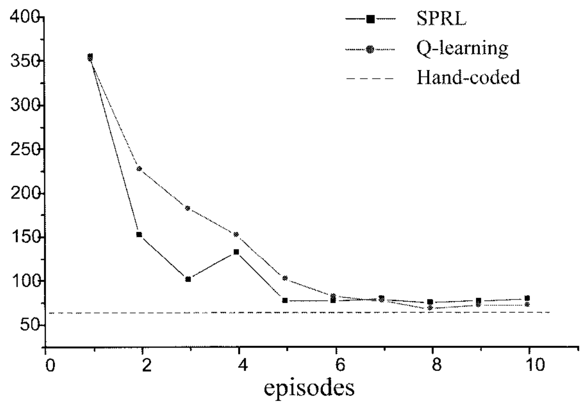


(a) Location 1. (b) Location 2.

Fig. 12. Two relative locations of a robot with respect to the box.

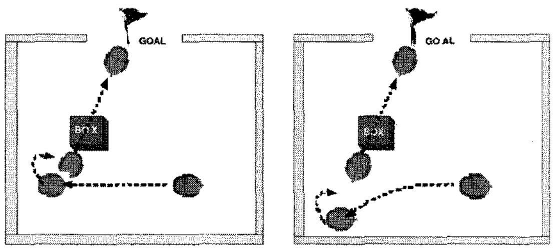


(a) The case when starting configuration of the robot, the box, the goal is given as shown in Fig. 12(a).



(b) The case when starting configuration of the robot, the box, the goal is given as shown in Fig. 12(b).

Fig. 13. The number of steps for PBIG task.



(a) Hand-coded. (b) SPRL learning.

Fig. 14. Trajectory of a robot.

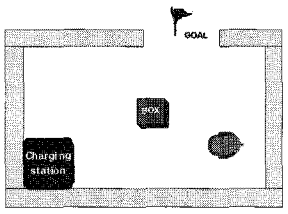


Fig. 15. Experimental setup for multi-task coordination.

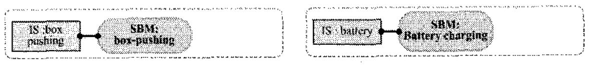
show reasonable performances for the PBIG task when compared to those by the other two methods.

A clear difference between hand-coded and learning methods is mainly observed in the learning of 'move-to-box' action pattern. From the action patterns obtained by learning experiments, combination of two primitive actions 'moveF' and 'moveFFL' defined as in Table 2 is often observed as in Fig. 14. It is noted that 'moveF' is observed as the primary actions. As the number of trials is increased, the performances of action patterns by hand-coded and learning methods are observed to be very similar.

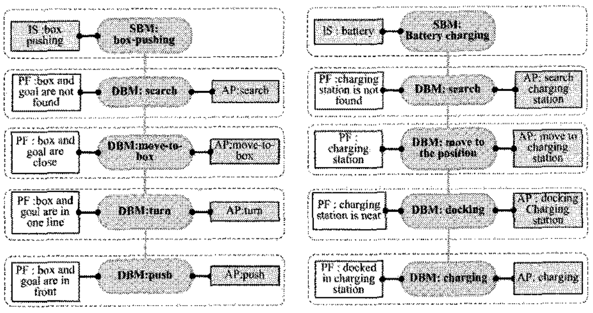
5.5. Learning of action patterns

After learning of action patterns is completed, another learning process can be activated to learn associations between perception filters and action patterns, and a sequence of these associations; this is known as learning of a sequence of DBMs.

A PBIG task and a battery-charging task are to be performed. Two corresponding SBMs are assumed to be innately defined. However, necessary DBMs are not attached to those two SBMs. Thus, DBMs should be learned to do two SBMs. Fig. 15 shows an experimental environment for the PBIG and battery charging tasks. The internal state value for PBIG SBM is given as a constant value, and internal state value for battery-charging is designed to become higher as battery voltage becomes low. When battery-voltage becomes low, the value of SBM being affected by internal state value for battery charging becomes high. Thus, SBM for battery-charging task is selected. But, when the battery-voltage is high, the value of SBM for battery-charging becomes low. Thus, SBM for the PBIG task is selected. After one of two SBMs is selected, learning is performed because there



(a) Before SPRL process.


IS :box pushing
SBM: box-pushing
PF :box and goal are not found
DBM: search
AP:search
PF : box and goal are close
DBM:move-to-box
AP:move-to-box
PF :box and goal are in one line
DBM:turn
AP:turn
PF : box and goal are in front
DBM:push
AP:push
IS : battery
SBM: Battery charging
PF :charging station is not found
DBM: search
AP: search charging station
PF : charging station
DBM: move to the position
AP: move to charging station
PF : charging station is near
DBM: docking
AP : docking Charging station
PF : docked in charging station
DBM: charging
AP: charging

(b) After SPRL process.

Fig. 16. DBM trees for two SBMs; pusing-box-into-a-goal and battery charging.

Table 4. Number of steps per episode(in case of 'move-to-box'AP).

episode	1	2	3	4	5
No. of steps	45	12	8	7	7

are no DBMs to achieve the SBM. Here, necessary action patterns are available, since they can be learned as shown in Sec. 5.4. SBM and DBM structures before and after learning are shown in Fig. 16(a) and (b), respectively. In Table 4, the number of action patterns is shown for each episode. It is observed from Table 4 that 45 steps are required for the first episode, but fortunately only 7 steps are shown to be necessary for the 4th and the 5th learning trials. Owing to our proposed SPRL method, learning of sequential behaviors is completed within a relatively small number of episodes.

It is remarked that success or failure of a task is heavily dependent not on the action selection method, but on other factors including the search strategy for learning, and perception accuracy. Therefore, there were difficulties to measure the performance of our proposed ASM. As one of our future works, long term performance needs to be measured for multiple SBMs for more practical tasks.

6. CONCLUSIONS

It is most important for a robot to select and learn the best behaviors to survive in an environment. To accomplish this, we proposed a hierarchical organization of competence modules called SBMs and DBMs. The SBM was used to select the most appropriate motivation in a given situation. The DBM was used to select a behavior that could satisfy its motivation. By utilizing releasers used to pass or block the activation-value of a DBM, a DBM tree can generate sequential behaviors. Thus, not only can our proposed ASM select the most appropriate behavior in a given situation, it can also deal with sequential behaviors. Furthermore, our proposed flexible tree-network enables the robot to effectively add learned behaviors.

REFERENCES

[1] A. Guillot and J.-A. Meyer, "Computer simulations of adaptive behavior in animals," *Proc. Computer Animation '94*, pp. 122-131, May 25-28, 1994.

[2] S. W. Wilson, "The animat path to AI," *Proc. of the First International Conference on Simulation of Adaptive Behavior on From Animals To Animats*, Cambridge, MA, USA, MIT Press, 1990.

[3] P. Maes, "Modeling adaptive autonomous agents," *Artificial Life*, vol. 1, no. 1-2, pp. 135-162, 1994.

[4] T. Tyrrell, *Computational Mechanisms for Action Selection*, Ph.D. Dissertation, University of Edinburg, 1993.

[5] M. J. Huber, S. Kumar, and D. McGee, "Toward a suite of performatives based upon joint intention theory," *Lecture Notes in Computer Science*, vol. 3396, pp. 226-241, 2005.

[6] J. Rosenblatt and D. Payton, "A fine-grained alternative to the subsumption architecture for mobile robot control," *Proc. of International Joint Conference on Neural Networks IJCNN*, pp. 317-323, 18-22 June 1989.

[7] E. Spier and D. McFarl, "A finer-grained motivational model of behavior sequencing," *Proc. of the Fourth International Conference on Simulation of Adaptive Behavior, From Animals to Animats 4*, 1997.

[8] P. Maes, "A bottom-up mechanism for behavior selection in an artificial creature," *Proc. of the First International Conference on Simulation of Adaptive Behavior on From Animals To Animats*, Cambridge, MA, USA, MIT Press, pp. 238-246, 1990.

[9] B. M. Blumberg, P. M. Todd, and P. Maes, "No bad dogs: Ethological lessons for learning in hamsterdam," *Proc. of the Fourth International Conference on the Simulation of Adaptive Behavior, From Animals to Animats*, 1996.

[10] Y. Hoshino, T. Takagi, U. Di Profio, and M. Fujita, "Behavior description and control using behavior module for personal robot," *Proc. of IEEE International Conference on Robotics and Automation*, vol. 4, pp. 4165-4171, Apr 26-May 1, 2004.

[11] T. Sawada, T. Takagi, and M. Fujita, "Behavior selection and motion modulation in emotionally grounded architecture for qrio sdr-4xii," *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, pp. 2514-2519, Sept. 28-Oct. 2, 2004.

[12] R. Brooks, "A robust layered control system for a mobile robot," *IEEE Trans. on Robotics and Automation*, vol. 2, no. 1, pp. 14-23, March 1986.

[13] E. de Sevin and D. Thalmann, "A motivational model of action selection for virtual humans," *Proc. of the Computer Graphics International*, Washington, DC, USA, IEEE Computer Society, pp. 213-220, 2005.

[14] M. Scheutz and V. Andronache, "Architectural mechanisms for dynamic changes of behavior selection strategies in behavior-based systems," *IEEE Trans. on Systems, Man and Cybernetics, Part B*, vol. 34, no. 6, pp. 2377-2395, Dec. 2004.

[15] C. M. Witkowski, *Schemes for Learning and Behavior: A New Expectancy Model*, Ph.D. Dissertation, University of London, 1997.

[16] M. Humphrys, *Action Selection Methods Using Reinforcement Learning*, Ph.D. Dissertation, University of Cambridge, 1996.

[17] S. Lee and I. H. Suh, "A programming framework supporting an ethology-based behavior control architecture," *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4138-4144, Oct. 2006.

[18] E. Hogewoning, J. Broekens, J. Eggermont, and E. G. P. Bovenkamp, "Strategies for affectcontrolled action-selection in Soar-RL," *Lecture Notes in Computer Science*, vol. 4528, pp. 501-510, 2007.

[19] R. Ameur and J.-C. Heudin, "Interactive intelligent agent architecture," *Proc. WI-IAT Workshops Web Intelligence and International Agent Technology Workshops IEEE/WIC/ACM International Conference on*, pp. 331-334, Dec. 2006.

[20] M. Azar, M. Ahmadabadi, A. Farahmand, and B. Araabi, "Learning to coordinate behaviors in soft behavior-based systems using reinforcement learning," *Proc. of International Joint Conference on Neural Networks*, pp. 241-248, 2006.

[21] N. Goerke and T. Henne, "Learning hierarchical action selection for an autonomous robot," *Proc. of International Joint Conference on Neural Network*, pp. 4958-4965, 16-21 July 2006.

[22] J. Qiao, Z. Hou, and X. Ruan, "Q-learning based on neural network in learning action selection of mobile robot," *Proc. of IEEE International Conference on Automation and Logistics*, pp. 263-267, 18-21 Aug. 2007.

[23] Z. Shen, C. Miao, Y. Miao, X. Tao, and R. Gay, "A goal-oriented approach to goal selection and action selection," *Proc. of IEEE International Conference on Fuzzy Systems*, pp. 114-121, 2006.

[24] J. Jaafar, E. McKenzie, and A. Smaill, "A fuzzy

action selection method for virtual agent navigation in unknown virtual environments," *Proc. of IEEE International Fuzzy Systems Conference*, pp. 1-6, 23-26 July 2007.

[25] P. Pirjanian, "Behavior coordination mechanisms-state-of-the-art," *Tech-report IRIS-99-375*, Institute for Robotics and Intelligent Systems, University of Southern California, Tech. Rep., 1999.

[26] J. H. Connell, "A behavior-based arm controller," *IEEE Trans. on Robotics and Automation*, vol. 5, no. 6, pp. 784-791, Dec. 1989.

[27] I. H. Suh, S. Lee, W. Y. Kwon, and Y.-J. Cho, "Learning of action patterns and reactive behavior plans via a novel two-layered ethology-based action selection mechanism," *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1799-1805, 2-6 Aug. 2005.

[28] H. S. Ahn and J. Y. Choi, "Emotional behavior decision model based on linear dynamic systems for intelligent service robots," *Proc. of the 16th IEEE International Symposium on Robot and Human interactive Communication*, pp. 786-791, 26-29 Aug. 2007.

[29] J. Bryson, "Hierarchy and sequence vs. full parallelism in action selection," *Proc. of the Sixth International Conference From Animals to Animats 6*, R. P. Jean-Arcady Meyer, Ed. MIT Press, 2000.

[30] C. L. Hull, *Principle of Behavior*, Appleton-Century-Crofts, New York, 1943.

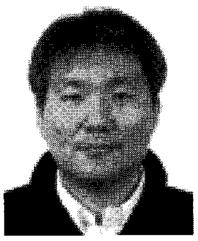
[31] W. Kwon, I. H. Suh, S. Lee, and Y.-J. Cho, "Fast reinforcement learning using stochastic shortest paths for a mobile robot," *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 82-87, Oct. 29-Nov. 2 2007.

[32] M. Kim and H. Kim, "Behavior coordination mechanism for intelligent home," *Proc. of the 5th IEEE/ACIS International Conference on Computer and Information Science*, pp. 452-457, 10-12 July 2006.

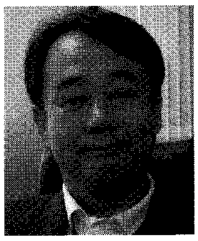
[33] R. Arkin, M. Fujita, T. Takagi, and R. Hasegawa, "Ethological modeling and architecture for an entertainment robot," *Proc. of IEEE International Conference on Robotics and Automation*, vol. 1, pp. 453-458, 2001.

[34] M. N. Nicolescu and M. J. Matari´c, "A hierarchical architecture for behavior-based robots," *Proc. of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, ACM, New York, NY, USA, pp. 227-233, 2002

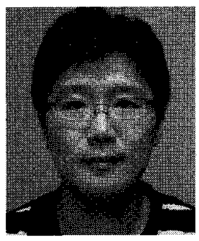
[35] A. Doucet, N. de Freitas and N. Gordon, *Sequential Monte Carlo Methods in Practice*, N. Gordon, Ed. Springer-Verlag, 2001.



Sanghoon Lee received the B.S. degree in 1994 from the Department of Mathematics, and the M.S. and Ph.D. degrees in Electronics, Electrical, Control, and Instrumentation Engineering from Hanyang University, Korea in 1997 and 2000, respectively. Currently, he is a Research Fellow at the Education Center for Network-based Intelligence Robotics, Hanyang University. His research interests include ethology-based action selection architecture, robot vision, and robot intelligence.



Il Hong Suh received the Ph.D. degree in Electrical Engineering from the Korea Institute of Science and Technology (KAIST), Seoul, in 1982. From 1985 to 2000, he was with department of EECS, Ansan campus, Hanyang University, where he was a Full Professor. From 2000 to present, he has been with the college of Information and Communications, Hanyang University, Seoul, Korea, where he is a Full Professor. And from August 2004 to July 2006, he also served as Dean for college of Information and Communications, Hanyang University, Seoul, Korea. He has served as the President of Systems and Control Society of Korea Institute of Telecommunications Engineers (2005-2007), and he is now President of Korea Robotics Society for 2008. He is also listed in Who's Who in the World, 2008. He has been involved in a number of Korea National Projects such as Intelligent Robotics Frontier Research Program for 21st Century, and URC (Ubiquitous Robotics Companion). He served as the Chair of steering committee for RUPI (Robot Unified Platform Initiative) organized by former Korean ministry of Information and communications for a period of 2006-2007. His research interests lie in the area of communications and intelligence for robots including web-based control of robots, ontology-based robot intelligence, context-adaptive action-coupled perception and learning, robot software platform. He has published more than 170 contributions in robotics and control.



Woo Young Kwon received the B.S. degree in 2001 in Mechanical Engineering at Hanyang University. He received the M.S. degree from the Department of Information and Communication at the same University in 2003. Currently, he is a Ph.D. student in the Department of Electronic Computer Engineering at Hanyang University. His research interests include action selection mechanism, robot software architecture, reinforcement learning, software agents, and robot intelligence.