Learning Attentive-Depth Switching while Interacting with an Agent

Chyon Hae Kim, Hiroshi Tsujino, and Hiroyuki Nakahara

***Abstract*— This paper addresses a learning system design for a robot based on an extended attention process. We consider that typical attention that consists of the position/area of a sight can be extended from the viewpoint of reinforcement learning (RL) systems. We propose an RL system that is based on extended attention. The proposed system learns to switch its attention depth according to the situations around the robot. We conducted two experiments to validate the proposed system: a capture task and a navigation task. In the capture task, the proposed system learned faster than traditional systems using switching. Q-value analysis confirmed that attention depth switching was developed in the proposed system. In the navigation task, the proposed system demonstrated faster learning in a more realistic environment. This attention switching provides faster learning for a wider class of RL systems.**

I. Introduction

Learning about human personalities and habits is an important area in the robotics field, because robots will come closer to humans in the future, serving in our houses and offices by utilizing their mobility. In such situations, robots will need to be able to learn and predict the behaviour of a human to improve their service.

There are three main requirements for a suitable learning system.

1) Adaptation to the tasks
2) Adaptation to humans
3) Fast adaptation

The robot's learning systems should adapt to human-related tasks, because the objective of a robot in many cases is to serve humans. Adaptation for humans is also important because the robots must be able to execute a task without receiving the required commands from the humans as far as possible, by estimating the requirement adaptively. Finally, fast adaptation is required to achieve sufficient performance in uncertain situations between a human and a robot.

For example, when a robot navigates a patient to a given position in a hospital, it should consider how to reach the required position and how to keep the human comfortable at the same time. This type of navigation requires the robot to learn the personality and habits of the patient through interaction. This learning must be sufficiently rapid, because the robot needs to adapt to the surrounding and changing environemnt.

Chyon Hae Kim and Hiroshi Tsujino are with Honda Research Institutes Japan Co., Ltd., 8-1 Honcho, Wako-shi, Saitama 351-0188, Japan
phone:+81-48-462-5219
E-mail:{tenkai, tsujino}@jp.honda-ri.com
Hiroyuki Nakahara is with Integrated Theoretical Neuroscience RIKEN Brain Science Institute, 2-1 Hirosawa Wako Saitama, 351-0198, Japan
phone:+81-48-467-9663
E-mail:hn@brain.riken.jp

A. Traditional Systems

Traditionally, reinforcement learning (RL) has been used to develop task-based learning systems [1], [2], [3]. An RL system allows robot creators to determine a robot's task by using a reward function. This distinguishes this type of learning system from others.

Multi-agent RL (MARL) [4], sub-category of RL, develops a relationship between self (robot) and an agent. For example, Tesauro proposed Hyper-Q based on the framework of MARL and demonstrated a rock-paper-scissors game [5]. While the agent plays this game with the robot, the robot must learn the personality and habits of the agent who is trying to win the game. However, disadvantage of MARL is that it requires the robot to learn large amount of data, resulting in slow learning speed because data sampling costs time for a robot.

In order to improve the speed, in this study, we focus on attention depth, which is based on extended attention. Typical studies of attention consider the position and area of a robot sight. Lucas et al. proposed an RL system that learns how to switch a robot's attention. The robot successfully decreased the entropy while recognizing objects in camera images [6]. Yoshikai et al. demonstrated a robot that imitates a human while shifting attention by using a learning system [7].

However, we consider that the concept of attention has been used in a restricted sence and can be extendable from the viewpoint of RL systems. In this paper, we explain this extended attention concept and propose an RL system that is equipped with several layers, which achieve attention switching in terms of extended attention.

The remainder of this paper is organized as follows, Section II describes the proposed system, Section III confirms the learning speed and attention switching of the system using a capture task, Section IV demonstrates the applicability of the system by using a navigation task, Section V discusses the performance and future development of the system. Finally, the conclusion is presented in Section VI.

II. Proposed System

A. Approach

Traditionally, attention has been defined as a robot sight directed to a certain position or area. In other words, attention is an observation selection function for a robot. We consider that the definition of attention can be extended to include the selection process for the state space of a robot from the viewpoint of RL.

For example, when a robot focuses its visual attention on an object, the robot observes information from the object selectively while neglecting other information than that inside

its attention area. Traditionally, the term attention has been used to refere to this type of process. After observation, the robot may filter the information to make its action selection easier. Especially when RL systems are applied, this type of filtering is required to form a sufficiently small state space because a large state space results in slow learning and low generalization of experience. We consider that the role of this filtering process is very similar to that of the traditional concept of attention. We have termed this process extended attention.

Extended attention will reduce the number of data samples required from RL systems because their learning speed positively correlates to the size of state space. However, exact implementation of extended attention is difficult for robot creators because they do not have sufficient information to anticipate the exact situations, in which a robot will interact with a human or an environment after development and distribution. Therefore, we try to implement extended attention as a result of a learning process of the robot.

B. Abstract Domain Definition

Before explaining the proposed learning system, we define the domain abstractly. In this domain, a robot interacts with an agent using its motion while proceeding with a task. The robot needs to observe self X, agent Y and environment Z information. For robotics researchers, considering the required state space for self X and environment Z information is relatively simple than that for agent Y information. This is because developers know self (robot) X information readily, and can describe environment Z information by using physical knowledge. However, agent Y information, which is dominated by the agent's thinking is very difficult to describe in robot systems.

Therefore, mechanical learning to select sufficient state space for information Y is important, this often cannot be predetermined by robot creators. In this paper, we introduce the concept of extended attention to the information Y and $\dot{Y}$.

C. Proposed System

We propose an RL system that learns attention switching regarding an agent's information by using competitive learning. After learning, the proposed system will select whether to neglect the velocity of agnet $\dot{Y}$ on the basis of the system's observation.

D. System

Figure 1 shows the proposed RL system, which uses two RL layers that are composed of finite states. The first layer (Layer1) has a state space (X, Y, Z) that does not include $\dot{Y}$. The second layer (Layer2) has a state space including $\dot{Y}$ $(X, Y, \dot{Y}, Z)$. When Y does not include absolute coordinate information, the system needs to predict the velocity $\dot{Y}$ using an estimator.

We need to design the state space of RL appropriately to accelerate its learning. A larger state space causes a slower learning speed because the system requires a larger number

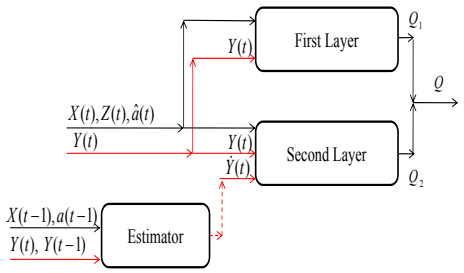

Fig. 1. Proposed reinforcement learning system

X: robot information. Y: agent information other than velocity. $\dot{Y}$: agent's velocity information. Z: environment information. a: action performed in the previous time step. $\hat{a}$: action that is considered to be performed.

of data to fix its function approximator. On the other hand, a smaller state space causes local optima of action selection learning because the state space has the potential to have less information to describe the selection rule. However, predetermination of size is very difficult for agent information Y and $\dot{Y}$, as described above.

Therefore, we introduce a type of competitive learning between the layers that selects whether $\dot{Y}$ (larger state space) is used or not (smaller state space). We define the states of the first and second layers as s_i and s_{ij}, respectively. The state of Layer 2 needs an additional index j because Layer 2 has a larger number of dimensions than Layer 1. When the state of Layer 2 is s_{ij}, the state of Layer 1 must be s_i, because these layers take a common observation from the sensory input $(X, Y, \dot{Y}, Z)$, although the state of Layer 1 lacks $\dot{Y}$. We define the Q-values of the first and second layers as $Q_1 := O_i(s_i)$ and $Q_2 := O_{ij}(s_{ij})$, respectively. If we define the total Q-value of these layers as $Q := \lambda_1 Q_1 + \lambda_2 Q_2$, then the Bellman error formulation is as follows:

$$E = \frac{1}{2} \sum_{i,j} \sum_{i',j'} p_{ij} P_{s_{ij}, s_{i'j'}} (r_{s_{ij}, s_{i'j'}} + \gamma Q'(s_{i'j'}, \pi) - \lambda_1 O_i - \lambda_2 O_{ij})^2 \quad (1)$$

where p_{ij} is the probability that Layer 2 takes state s_{ij}, $P_{s_{ij}, s_{i'j'}}$ is the probability that a robot transits from the state s_{ij} to another state $s_{i'j'}$ using its action selection policy π, $r_{s_{ij}, s_{i'j'}}$ is rewarded during the transition, and Q' is the Q-value of the selected action based on the policy π and the state $s_{i'j'}$.

To deduce the update function for each O, we assume the terms $r_{s_{ij}, s_{i'j'}} + \gamma Q'(s_{i'j'}, \pi)$ to be the output target value of the system that is independent of O_i and O_{ij}. Under this assumption, we apply the steepest descent method.

$$\Delta O_m = -\alpha \lambda_1 \frac{\partial E}{\partial O_m} \approx \alpha \sum_j \sum_{i',j'} p_{mj} P_{s_{mj}, s_{i'j'}} (r_{s_{mj}, s_{i'j'}} + \gamma Q'(s_{i'j'}, \pi) - \lambda_1 O_m - \lambda_2 O_{mj}) \quad (2)$$

$$\Delta O_{mn} = -\alpha \frac{\partial E}{\partial O_{mn}} \approx \alpha \lambda_2 \sum_{i',j'} p_{mn} P_{s_{mn}, s_{i'j'}} (r_{s_{mn}, s_{i'j'}} + \gamma Q'(s_{i'j'}, \pi) - \lambda_1 O_m - \lambda_2 O_{mn}) \quad (3)$$

This method gives these offline update functions. The online versions of the functions are as follows:

$$\Delta Q_1 = \alpha_1(r + \gamma Q' - \lambda_1 Q_1 - \lambda_2 Q_2) \quad (4)$$

$$\Delta Q_2 = \alpha_2(r + \gamma Q' - \lambda_1 Q_1 - \lambda_2 Q_2) \quad (5)$$

These update functions realize a type of competitive learning. Q_1 and Q_2 inhibit the increasion of Q_2 and Q_1 each other because update values ΔQ_1 and ΔQ_2 consist of $-\lambda_2 Q_2$ and $-\lambda_1 Q_1$, respectively. In particular when $\lambda_1 = \lambda_2 = 1$, Q_1 and Q_2 are balancing with equal rates. To increase Q_1 or Q_2 with keeping stable, we need to decrease Q_2 or Q_1 by the same amount. When we set one of the α_is to 0, the functions work in the same way as a traditional reinforcement learning, SARSA. In this paper, we update each Q_i by using these online update functions. Moreover, we use mesh-type function approximators for the approximation of the Q values and the ϵ-greedy method for the action selection.

III. Capture Task

We conducted a capture task to validate the learning speed and attentive level switch of the proposed system. A robot that is represented by a half circle captures the centre of the mass of an agent in this abstracted task (Fig.2).

In general, there are two methods by which a robot can capture an agent. The first method is to use a feedback control for the position Y of the agent. This method is optimal only when the agent does not move or when the agent moves randomly. The second method is to estimate the agent's motion from its position and velocity $(Y, \dot{Y})$. When the agent has some rules in the motion, this method will work better than the former method, although the robot needs to learn the rules.

A. Experimental Settings

We set the half circle radius of the robot $R = 0.1\ [m]$. For the vertical direction, the robot approaches an agent with a constant velocity $v = 0.2\ [m/s]$. For the horizontal direction, the robot selects its action among the following three actions: moving the half circle to the left by the length of $\Delta = 0.2\ [m]$, moving the half circle to the right by the length of Δ, and remaining still. The robot gets a reward of 1 or -1 when it succeeds or fails, respectively, to capture the agent.

We implemented a type of randomness and a rule for the agent. The agent shifts its position p left and right using a normal random number $\phi(u, \sigma^2)$ every Δt seconds.

$$p(t + \Delta t) = p(t) + \phi(u, \sigma^2) \quad (6)$$

If u and σ^2 are both constants, the agent's motion is a completely random walk. We then introduce a hidden rule inside the random walk. When the sign of u changes periodically ($u = u_0\ or\ -u_0$), the agent's motion is slightly different from random. There is an inhibited tendency to go left/right at a given moment. If a robot is able to estimate u, it might increase its capture rate.

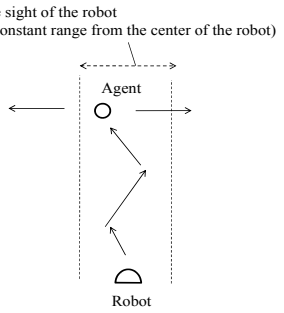

Fig. 2. Simulated capture experiment

The estimation difficulty can be controlled by changing the parameter σ^2. A larger or smaller σ^2 increases or decreases the randomness of the motion, respectively.

The agent's motion flow is as follows:

{1} Decide the parameters u_0 and σ.
{2} Add normal random number $\phi(u, \sigma^2)$ to p (this process repeats n times).
{3} Invert the sign of u.
{4} Continue Steps 2 and 3.

We set the initial position of the agent to be just 1 [m] above the centre position of the robot. The robot continues to learn for one set (equal to 20,000 trials) using the same parameters $\Delta = 0.2\ [m]$, $u = 0.2\ [m]$, $\sigma^2 = 0.15\ [m^2]$, $\Delta t = 0.2\ [s]$, and $n = 1$. At the start of the learning process, we initialized all Os of Layer1 and Layer2 to zero. We used an optimistic action selection [3] by adding bias to the Q value ($Q = Q_1 + Q_2 + 0.007$). We set the learning rate α of each layer to 0.08.

For the state Y of Layer1, we used the relative position vector $(\Delta x, \Delta y)$ from the mass centre of the robot to the agent. For the input $\dot{Y}$ of Layer2, we used the horizontal absolute velocity of the agent $\dot{y}$.

In order to compare the performance, we used two SARSAs that correspond to the layers. We fixed the parameters of the SARSAs to correspond to those layers including the learning rate, 0.08.

B. Results

1) Success Rate: We performed 100 sets of experiments for each system. Fig. 3(left) shows a 2,000 trial moving average of the capture rates for the 100 sets with standard deviation error bars Fig. 3(right) shows a 200 trial moving average to show the detail. These graphs show that the proposed system could achieve a higher success rate than the traditional systems, SARSA1 and SARSA2, which correspond to the first and second layers, respectively. The dotted lines in Fig. 3(left) show reference performances that represent the best performances of ideal systems. The robot of Reference 1 does not know the sign of u and follows the feedback control to $y(t)$. This reference is optimal when u does not change its sign. The robot of Reference 2 knows the sign of u and follows the feedback control to $y(t) + u$. This reference shows the maximum capture rate when the robot predicts the agent's motion completely.

2) Analysis of Attention: In order to analyze the attentive level of the proposed system, we focused on the domination

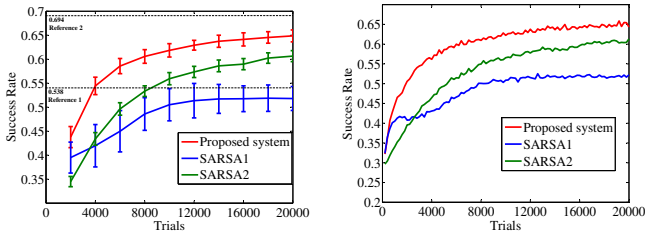

Fig. 3. Success rate (left/right side figure shows 2,000/200 moving average)

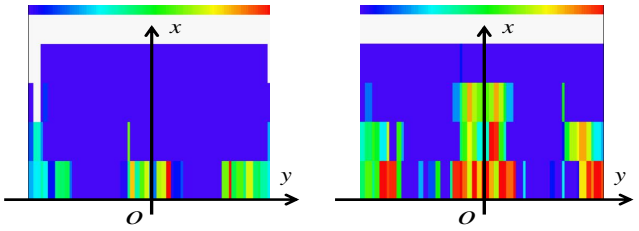

Fig. 4. Attention during the early stage (left) and that during the last stage (right)

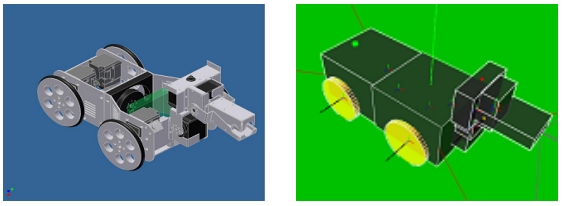

Fig. 5. Mechanical model (left) and computational model (right)

of the layers. When Layer 1 dominates the action of the robot, the robot moves on the basis of only input Y because Layer 1 does not obtain input $\dot{Y}$. This means that the robot's attentive level is shallow. As mentioned above, in this attentive level, the robot's best strategy is to follow $y(t)$ (Reference 1), because the robot has no way to estimate u from information Y. When Layer 2 dominates the action of the robot, the robot moves on the basis of input $(y, \dot{y})$. In such a case, the robot's attentive level is deep, and the best strategy is to follow $y(t) + u$. Therefore, dominance of the layers is an effective tool to validate the attentive level of the layers.

We analysed the attention level switch between Y and $(Y, \dot{Y})$ while the robot was performing the task by using the dominance. We defined dominance of Layer 1, D_1, as follows:

$$D_1 = \sum_{\dot{y}} \delta_{a(\Delta x, \Delta y, \dot{y}), a_1(\Delta x, \Delta y)} \tag{7}$$

where δ is a Kronecker delta, a is an action that the whole system selects as best one based on Q value, and a_1 is an action that Layer 1 selects as the best one on the basis of the Q_1 value while neglecting Q_2. In this definition, when D_1 is high or low, the robot's action is dominated by the first or second layer, respectively.

The left and right sides of Fig. 4 show the attention level of the robot on the basis of the dominance definition. Each coloured pixel of these images shows the dominances of Layer 1 D_1 while the robot learns the capture task. We set O as the centre position of the robot. In the colour bar at the top of the images, blue indicates that the dominance of Layer 1 is strong (D_1 is high) and red indicates that dominance of Layer 1 is weak (D_1 is low). At an early stage of learning, Layer 1 dominance was strong in a wide area. This means that the robot had attention for $(\Delta x, \Delta y)$ and ignored the velocity $\dot{y}$ of the agent. During the final stage of learning, dominance of Layer 1 is weak at the centre, on the left, and on the right. This means that the agent was near the robot or around the limitation of the sight of the robot, the robot focused on the motion that includes velocity $\dot{y}$. While the agent was far from the robot, the robot still focused on $(\Delta x, \Delta y)$. Therefore, the final stage dominance shows the attentive-level switch inside the learning system according to the related position $(\Delta x, \Delta y)$ from the robot to the agent.

IV. Navigation Task

We applied the proposed system for a navigation task. In this task, a navigation robot learns how to navigate another robot to a goal area. In previous studies, several researchers have attempted such navigation tasks using traditional systems such as a control system using a potential field [8], [9], an evolutionary computation system [10], and a classifier system [11]. Vaughan's control system gathered a flock of animals at a point by using a feedback control [8], [9]. However, Vaughan did not consider the implicit rules of the flock. This is the same as the situation in which soccer robots do not consider the implicit rules of a soccer ball [12] other than its physical dynamics.

We consider that this task is suitable for validation of the proposed learning system because the quantitative performance validation is easy and reproducibility is higher than when a human acts as a navigated agent. We analysed the proposed system using this task.

We designed the simulation model of the robots from the hardware model (Fig. 5 right) and designed the environment (Fig. 6) on the Webots simulator [13]. Table I shows the specifications of the robots. We used the same model for both the guiding robot and the guided robot.

A. Guiding Robot

The software system of the guiding robot has three components: a pre-processing system, a learning system, and a behavior-generation system.

1) Pre-processing System: In this system, the information obtained from the head-mounted camera image and the encoder is processed, and the result is sent to the learning system (Table II). From the image obtained by the head-

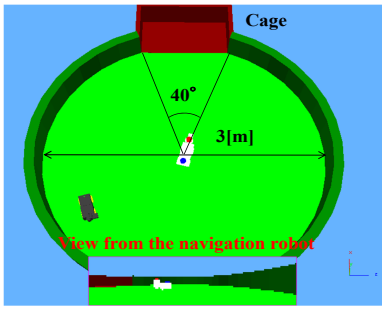

Fig. 6. Experimental environment

TABLE I

SPECIFICATIONS OF THE GUIDING AND GUIDED ROBOTS

Weight	Head		184.7 [g]
	Body (front)		370 [g]
	Body (back)		300 [g]
Size	Body	Width	120 [mm]
		Length	250 [mm]
	Wheels	Width	10 [mm]
		Radius	40 [mm]
DOF	Track Wheels		(D.O.F = 2)
	Waist		Roll (1)
	Neck		Pitch and Yaw (2)
	Jaw		Raises and lowers snout of robot (1)
Devices	Camera	Field of View	2 radians
		Resolution	64×64 pix.
	IR Sensor	Quantity	4
		Placement	30 degrees from side parallel
	Gyroscope		(not in the model)

TABLE II

STATE SPACE OF THE LEARNING SYSTEM

Target	Information (dimensions)	Range
Self (x)	Neck yaw (1)	$[0, 1]$
Other agent (y)	Horizontal weight center (1) Rotation ($cos\theta$, $sin\theta$) (2)	$[0, 1]$ (detected) -1 (not detected)
Cage (z)	Horizontal weight center (1) Horizontal corner position (2)	$[0, 1]$ (detected) -1 (not detected)

mounted camera, the guiding robot extracts the weight centres of the guided robot, the cage, and the LEDs on the guided robot using the thresholds of their colours. The guiding robot then calculates the direction of the guided robot from the position of the LEDs. In addition, the guiding robot extracts the vertical edges of the cage by using Hough transform. The horizontal weight centres of the guided robot and the cage, the sine and cosine of the direction vector of the guided robot, and the horizontal positions of the edges of the cage are normalized to the range of [0,1]. If the objects are out of view and the guiding robot fails to detect the objects, -1 is assigned to the value of the information. The angle of the neck of the guiding robot obtained from the encoder is also normalized to the range of [0,1].

2) Learning System: We applied the proposed learning system (Fig. 1) with an estimator that is created through pre-learning of the robot.

In the capture task, we could calculate the velocity of the other agent $\dot{y}$ easily because $\dot{y}$ was always detectable without noise. In this guiding task, the calculation of $\dot{y}$ is difficult because image processing is not accurate and the positions of the objects are not always detectable. In such cases, the robot must learn to obtain an accurate estimator for calculating the predicted state of the other agent.

We constructed the estimator using an online learning process that has a mesh type function approximator. Each cell of the mesh outputs each prediction of $\tilde{y}$ for the corresponding state of the cell. The estimator calculates an average from the training data for $\tilde{y}$, and fixes the output of each cell to the average. We allowed the guiding robot to move randomly using its action primitives (described in the following subsection) around the guided robot in the experimental environment to update the estimator. This updating process continued for a simulation time of 10 hours.

We set several rewarding rules for the learning system according to the state of the robots. The learning system rewarded automatically with a reward of 0.1 when the guided robot and the cage overlapped on the image obtained by the head-mounted camera of the guiding robot. From this state, if the guiding robot moved towards the guided robot, the learning system received a reward of 1. When the guiding robot successfully completed the guidance and the guiding robot could confirm this on the image captured by the head-mounted camera, the learning system received a reward of 10.

TABLE III

ACTION PRIMITIVES

Index	Time [s]	Motion
A_0	1	Stay
A_1	1	Move toward the position of the target
A_2	2	Turn clockwise around the target
A_3	2	Turn counterclockwise around the target
A_4	1	Move away from the position of the target
A_5	1	Search for the target
A_6	5	Move away from the cage
A_7	1	Search for the cage

3) Action Primitives: We prepared eight action primitives (Table III). The guiding robot executed one of the primitives selected by its learning system.

B. Guided Robot

The guided robot moves according to its input from the infra-red (IR) sensors and the force field in the environment. The guided robot avoids obstacles in the field and the guiding robot using its IR-sensors (Table IV). This avoidance has a higher priority than movements that are according to the force field.

TABLE IV

COLLISION AVOIDANCE

Which sensors detect the objects	Command
Two front sensors	Turn left or right at random
Two rear sensors	Move forward
Right sensor only	Turn left
Left sensor only	Turn right

The guided robot follows the force field when nothing is detected by the IR-sensors. When the guiding robot is outside the 0.15 [m] radius circle around the guided robot, the guided robot follows the force field shown in Fig. 7 (left) and moves to the centre of the field. If the guiding robot is within the circle, then this force field changes its flow, as shown in Fig. 7 (right). Fig. 7 (right) shows the force field when the guiding robot approaches the guided robot from the downward direction from the figure. Even if the relative positions of the robots are the same, the guided robot moves differently on the basis of its position in the field.

C. Results

We compared SARSA1, SARSA2, and the proposed system using the same learning parameters as those used in the capture task. The success rate of the proposed system tended to be higher than those of the other systems (Fig. 8).

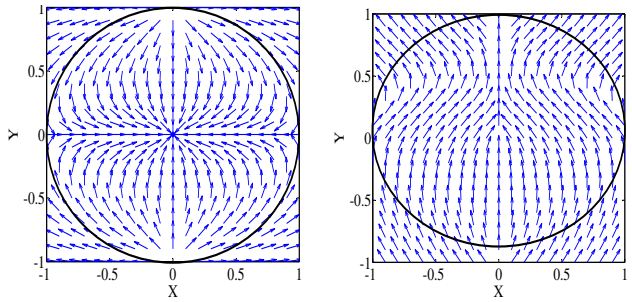

Fig. 7. Force field (left) and force field while avoiding the guiding robot (right)

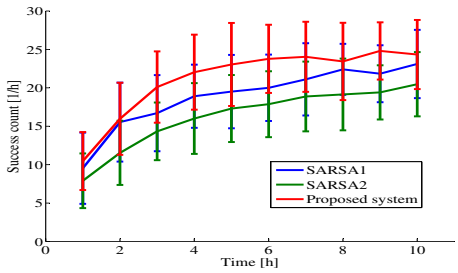

Fig. 8. Success rate of the guiding robot simulation

V. Discussion

A. Attentive Level Switching

The capture task analysed the attentive-level switching of the proposed system. We found that learning speed and attention have a strong relationship in RL systems.

In an early stage of learning the proposed system used only one attention. However, by the final stage, the proposed system switched to two attentions Y and $(Y, \dot{Y})$ according to the observation. This result shows that the proposed system had learned a type of attentive-level switching. The comparison between the proposed system and SARSAs shows the effectiveness of the attentive-level switch. The learning of the proposed system was faster than that of SARSAs.

The proposed system only switched between two types of attentions, however, the learning speed was dramatically improved. We need to investigate the number of types of attentions required to improve the speed.

B. Generalization

The attention depth of the extended attention concept is a highly general concept. In this paper, we focused on the velocity of an agent. However, this concept might be applicable to any RL that has number of choicees for state space. When we apply the concept to a learning system, we may need to arrange additional systems such as the estimator of the proposed system. Further research is required to develop a general framework to select the additional systems.

C. Validation Method

The validation method for the proposed system requires improvement. There is a trade-off between analysis capability and applicability validation. A simple numerical simulation, the capturing task, allowed us to validate the attention switch. However, this was not a realistic task. The navigation task was more realistic, however, this made validation difficult, because the amount of information required to analyse was so large. If we introduce a human as a guided agent, this task will be even more complicated. We, therefore, need to consider how to satisfy both analysis capability and applicability.

D. Learning System Topology

This research also showed that the network topology of RL systems (e.g. layers of the proposed system) is important to describe attention switching. Therefore, adaptation for the network topology [14], [15] is important to enhance the capability of the proposed system.

VI. Conclusion

In this paper, we proposed a learning system that learns the attentive level switch according to the state of agents. The results of the capture task simulation revealed that the capture rate of the proposed system was higher than those of the traditional methods. While learning, the proposed system learned attentive-level switch. The results of the guiding task simulation showed a higher success rate than traditional methods in a more realistic task.

References

[1] C. J. C. H. Watkins: ”Learning From Delayed Reward,” Ph.D. thesis of Cambridge University, (1989).
[2] C. J. C. H. Watkins:”Q-Learning,”Machine Learning, Vol. 8, pp. 279–292, (1992).
[3] R. S. Sutton and A. G. Barto: ”Reinforcement Learning,”MIT Press, 55 Hayward Street Cambridge, MA 02142–1493 USA, (2000).
[4] E. Yang and D. Gu: ”Multiagent Reinforcement Learning for Multi-Robot Systems: A Survey,” University of Essex Technical Report, (2004).
[5] G. Tesauro: ”Extending Q-Learning to General Adaptive Multi-Agent Systems,”Advances in Neural Information Processing Systems, (2003).
[6] L. Paletta, G. Fritz, and C. Seifert: ”Reinforcement Learning of Informative Attention Patterns for Object Recognition,”Proceedings of IEEE International Conference on Development and Learning, (2005).
[7] T. Yoshikai, N. Otake, and I. Miznuchi: ”Development of an Imitation Behavior in Humanoid Kenta with Reinforcement Learning Algorithm Based on the Attention during Imitation,” Proceedings of IEEE/RSJ International Conference on intelligent Robots and Systems, (2004).
[8] R. Vaughan, N. Sumpter, A. Frost, and S. Cameron: ”Robot Sheepdog Project Achieves Automatic Flock Control,” Proc. of the International Conference on Simulation of Adaptive Behavior, (1998).
[9] R. Vaughan, N. Sumpter, J. Henderson, A. Frost, and S. Cameron: ” Experiments in Automatic Flock Control,” Robotics and Autonomous Systems, Vol. 31, pp. 109–117, (2000).
[10] A. C. Schultz, J. J. Grefenstette, and W. Adams: ” RoboShepherd: Learning a Complex Behavior,” In Proceedings of the Robots and Learning Workshop, pp. 105–113, (1996).
[11] O. Sigaud and P. Gérard: ”Using Classifier Systems as Adaptive Expert Systems for Control,” Lecture Notes in Computer Science, pp. 138–157, (2000).
[12] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda: ” Purposive Behavior Acquisition for a Real Robot by Vision-based Reinforcement Learning,” Machine Learning, Vol. 23, pp.279–303, (1996).
[13] http://www.cyberbotics.com/
[14] K. O. Stanley and R. Miikkulainen, “Efficient Reinforcement Learning Through Evolving Neural Network Topologies,” In Proceedings of the Genetic and Evolutionary Computation Conference, (2002).
[15] C. H. Kim, T. Ogata, and S. Sugano: “Reinforcement Signal Propagation Algorithm for Logic Circuit”, Journal of Robotics and Mechatronics, (2008).