UNSYS digital

The Journal of Instrumentation, Automation and Systems

Automatic Convergence Estimation by Utilizing Fractal Dimensional Analysis for Reinforcement Learning

Hitoshi Kono[†], Tsuyoshi Suzuki[§], Akiya Kamimura[‡], Kohji Tomita[‡], Yusuke Tamura[δ], Atsushi Yamashita[δ], Hajime Asama[δ]

[†]The University of Tokyo (present affiliation is Tokyo Polytechnic University), Japan
[§]Tokyo Denki University, Japan
[‡]National Institute of Advanced Industrial Science and Technology (AIST), Japan
[δ]The University of Tokyo, Japan

Abstract—This paper presents an automatic convergence estimation method for the reinforcement learning (RL) of autonomous agents. Recently, a multi-agent robot system (MARS) that uses an RL algorithm has been studied in real-world situations. This system can obtain behaviors autonomously through multi-agent reinforcement learning (MARL). However, MARL takes a long time to obtain an optimal or near-optimal solution. Furthermore, if the agents continue learning after the solution is obtained, this may lead to overfitting. It is difficult for the MARL operator to determine whether the learning has been converged because the operator has to determine the convergence of all learning agents. The convergence of the learning curve indicates that knowledge has been obtained. However, judging the convergence of RL depends on human intuition and experience, and few studies have discussed convergence estimation methods. In prior work, an automatic convergence estimation method using fractal dimensional analysis that evaluates the learning curve of the learner as agents was proposed. This method did not require human judgment, and its effectiveness was confirmed by conducting only a computer simulation. In this study, we propose a method based on fractal dimensional analysis considering implementation of the robots. We evaluate the effectiveness of the method by using an expanded experimental setup with a computer simulation and an actual mobile robot environment. The main contribution of this paper is verifying the effectiveness of the convergence estimation method through computer simulations and an experiment using an actual mobile robot.

Keywords—Automatic convergence estimation, fractal dimensional analysis, reinforcement learning, autonomous agent.

DOI: 10.21535/jias.v3i3.934

I. Introduction

Actual multi-agent robot systems (MARSs) have recently been deployed in real-world situations such as autonomous multi-robot conveyance systems in warehouses [1]. Other applications include a multi-robot inspection system for disaster-stricken areas and autonomous multi-robot security systems [2], [3]. However, the real world, in which MARSs must operate, is a dynamic environment. This complicates the development of the systems because developers must customize the robots to adapt to this dynamic environment. Multi-agent environments and human-robot symbiosis environments are examples of dynamic environments. One approach used to solve this problem is the application of multi-agent reinforcement learning (MARL) to MARSs. MARL is a mechanism for implementing a posteriori cooperation among agents, which can behave adaptively in a dynamic environment even when they are not provided with specific control policies and plans. The benefits of MARL have been demonstrated in various studies over the past decade [4]–[7]. However, it is difficult to apply MARL to MARS in real environments because a significantly long learning time is required. Moreover, a MARS typically contains at least one preprogrammed robot, and MARL suffers from the following drawbacks:

- The learning process requires a long time in actual environments and for the robots.
- The knowledge obtained depends on the situation (*e.g.*, dynamic or static environment, the number of agents, and locations of obstacles).
- The robot has limited capacity to store knowledge.

By contrast, the transfer learning (TL) technique can be used to overcome these drawbacks. TL is a framework for reusing the knowledge obtained through reinforcement learning (RL) [8]. Thus, the learning time is reduced by using the obtained knowledge, and an agent with TL can use the knowledge obtained from recent tasks for new tasks. We proposed a knowledge co-creation framework that expands the concept of TL [9] (**Figure 1**). Knowledge co-creation combines MARS, MARL, and TL. This concept is similar to cloud robotics [10], [11], which involves abstract concepts. However, our proposed concept focuses on a concrete knowledge sharing methodology utilizing a computer network. In prior research, we highlighted and investigated the importance of the following issues: "*Who transfers knowledge?*," "*When should knowledge be transferred?*," and "*How can knowledge be*

Corresponding author: Hitoshi Kono (e-mail: kono@robot.t.u-tokyo.ac.jp).
This paper was submitted on February 17, 2017; revised on May 21, 2017; and accepted on July 4, 2017; available online on September 26, 2017.

transferred?." In this paper, we focus on the "*When*" part of knowledge co-creation.

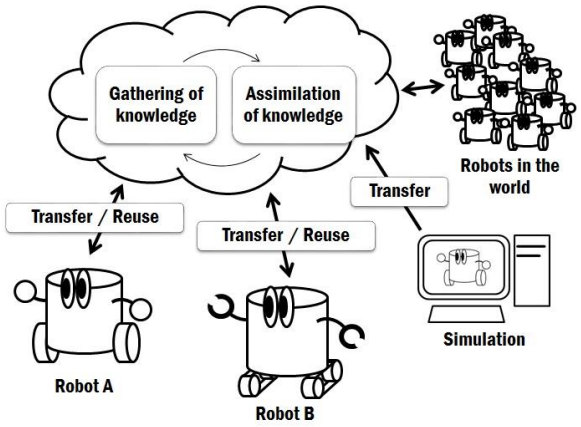


Figure 1 Simplified overview of our concept of the *knowledge co-creation* by the multiple learning robots

Intuitively, when an agent transfers knowledge to another agent that behaves using the obtained knowledge in a new environment, knowledge that is not fully learned is unsuitable for reuse. Therefore, before transferring the knowledge, the agent has to judge when the learning is finished. Typically, we can estimate the end of learning using a learning curve and its convergence. This estimation is conducted using human intuition and experience. Therefore, an agent needs an automatic convergence estimation method in which a learning curve is used to automatically judge the end of learning. In Ref. [12], Kono *et al*. discussed and developed an automatic convergence estimation method, and Kono et al. focused on the shape of the learning curve. In this previous work, Kono *et al*. found that the learning curve probably has a fractal shape. An automatic convergence estimation method using the fractal dimensional analysis method was proposed and evaluated through a computer simulation. In this paper, first we confirm that the previous method enables the estimation of the learning curves' convergence by conducting computer simulations. Second, we describe the extended method, which is the automatic convergence estimation method for implementation of the robot, and we confirm that the method enables the estimation of the learning curve's convergence by an actual experiment with a mobile robot. The main contributions of this paper are the evaluation and confirmation of its effectiveness, which are conducted through an actual robot experiment.

The remainder of this paper is organized as follows: Section II describes the basic theory and relevant research studies such as learning mechanisms and the estimation method of the convergence. Section III presents an overview of fractals, their calculation method, and the proposed automatic convergence estimation method. Section IV provides a basic evaluation of the effectiveness of the proposed method through computer simulations. Section V describes the evaluation of the effectiveness of the proposed method through an actual mobile robot experiment, and discusses the results. Finally, Section VI presents concluding remarks.

II. Basic Theory and Relevant Research

A. Reinforcement Learning

RL is a type of machine learning method in which agents use a trial-and-error method to create their own policies. Many types of RL mechanisms have been proposed over the past several decades. In this study, we adopt Q-learning [13] as the RL mechanism, defined as follows:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\{r + \gamma V(s') - Q(s,a)\}, \quad (1)$$

$$V(s) = \max_{a \in A}(s,a). \quad (2)$$

Here, S is a state space, with $s, s' \in S$; a is an element of an action space A; $\alpha (0 < \alpha \leq 1)$ is the learning rate; $\gamma (0 < \gamma \leq 1)$ is the discount rate; and r is the reward. For Q-learning, it has been proven that the obtained solution converges to an optimal solution in a Markov decision process (MDP) environment. The learning agents select one of the predefined actions a with a probability given by the Boltzmann distribution according to

$$p(a \mid s) = \frac{\exp\left(\frac{Q(s,a)}{T}\right)}{\sum_{b \in A} \exp\left(\frac{Q(s,a)}{T}\right)}. \quad (3)$$

Here, T is a parameter that determines the randomness of the selection. The Q-learning model can select actions in descending order according to the action value from the learned knowledge. When the values of the available actions are the same or equal to the default value, the Boltzmann distribution is used to select an action at random.

To evaluate the learning performance, in this paper, we adopt a learning curve.

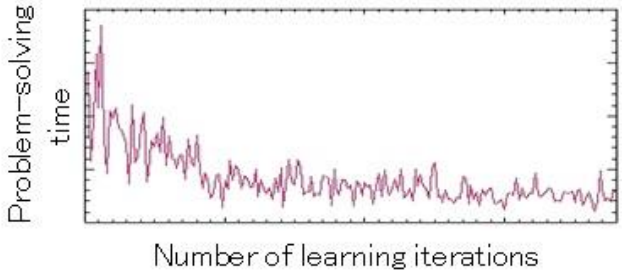


Figure 2 Example of a learning curve. The vertical axis indicates the problem-solving time of a learner, that is, the performance of the learner. The horizontal axis indicates the number of time or iterations of learning.

B. Learning Curve in RL

The learning curve is a metric for evaluating the performance and progress of learning. In general, a learning curve, as shown in **Figure 2**, is used in several disciplines such as pedagogics and engineering. In the RL domain, the learning curve is used to evaluate the performance of the learner and the progress of learning. We also use the learning curve as the output data of RL, which makes it easy to understand the state of learning.

Although Watkins *et al*. proved the convergence of Q-learning in an idealized environment modeled as an MDP, in which a common rule for convergence has not been explicitly defined [13]. If the problem to which RL is applied is simple, the solution is more likely to converge, and the obtained solution is either an optimal or a near-optimal solution. However, if the agent is in a dynamic environment, the agent learns many near-optimal solutions rather than a single optimal one.

Note that obtaining an optimal solution is not guaranteed when the environment is not MDP (*e.g.*, non-MDP or partially observable MDP). Therefore, it is difficult to define the relationship between convergence and obtaining the optimal solution when we assume that the environment is not MDP, such as an actual environment.

C. Convergence Test without RL Domain

In the domains of mathematics and physics, the theory of convergence testing has been studied. For example, the $\varepsilon - \delta$ limit definition and d'Alembert test are well-known approaches for judging the convergence of a function, and these test methods are similar to our convergence estimation. However, these test methods are oriented toward theoretical discussions, and an arbitrary value (*e.g.*, ε) is a very strict criterion for judging the convergence.

In a paper by Sichao *et al.* [14], the convergence was judged visually by a human. Visual judgment seems the most simple and intuitive method; however, human intuition and experience are not quantitative. Therefore, visual judgment is not easy to apply to other domains.

Otani *et al*. evaluated the convergence of behaviors of multiple robots using a relative error of 5% as the threshold value [15]. However, the static threshold value depends on the environment and system setup.

As previously described, existing methods have some drawbacks. For example, the learning agent needs a human observer and a threshold value that depends on the environmental settings.

D. Convergence Estimation in RL Domain

As mentioned before, it is difficult to define the relationship between the convergence of the learning curve and obtaining the optimal solution. However, in a real environment, the learning curve in most cases does not reach an optimal solution. Therefore, finding a convergence estimation method can contribute to making decisions by agents for acquiring a high-performance policy.

Kono *et al*. discussed and developed the automatic convergence estimation method in reinforcement learning [12]. This method judges the convergence of reinforcement learning based on the shape of its learning curve. However, Kono *et al*. evaluated the effectiveness of that method with only computer simulations. Kono *et al*. were unable to perform implementation into an actual learning system such as a mobile robot. Thus, for the implementation of a learning robot, convergence estimation methods have not been discussed in detail yet. Therefore, to realize autonomous knowledge sharing and transfer by agents, the agent needs an automatic convergence estimation method considering a learning robot.

III. Proposed Method

We focus on the shape of the learning curve as with Ref. [12], and we assume that the learning curve has a fractal shape. In this paper, we describe a convergence estimation methodology that uses fractal dimensional analysis.

A. Fractal and Fractal Dimension

A fractal, as proposed by Mandelbrot, is a shape that exhibits self-similarity. Fractals are characterized by a fractal dimension, which is a representation of how closely the shape fills the space it occupies [16]. Nature provides many examples of shapes exhibiting fractal characteristics (e.g., coastlines, snowflakes, and fern leaves). The fractal dimension is the numerical value of fractality. It can be used for classifying and evaluating the similarity of the observed target. An evaluation method that uses the fractal dimension is called fractal dimensional analysis.

B. Fractal Dimensional Analysis

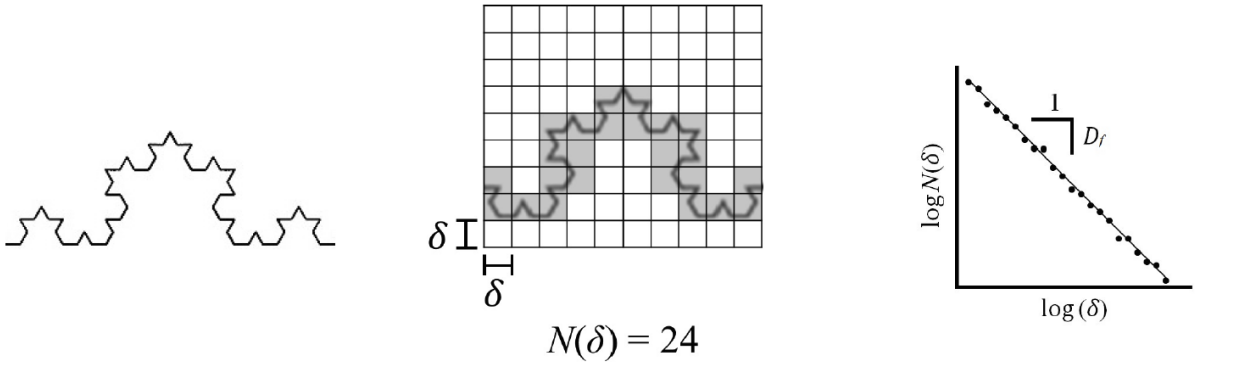
One of the methods for computing the fractal dimension uses the concept of box-counting dimension, which is easy to implement in a computer program. The box-counting dimension, D_f, is obtained as follows:

$$D_f = -\lim_{\delta \to 0} \frac{\log N(\delta)}{\log \delta}, \tag{4}$$

$$N(\delta) \sim c\delta^{-D_f}. \tag{5}$$

Here, $N(\delta)$ is the total number of squares necessary to fill the observation target with squares of side length δ. For example, **Figure 3 (a)–(c)** shows the calculation sequence of the fractal dimension using the box-counting method. The observed target (**Figure 3(a)**) is filled with squares of side length δ, as shown in **Figure 3(b)**. Then, overlapping squares of side length δ are counted, and D_f is obtained by calculating the regression slope of a double logarithmic plot, as shown in **Figure 3(c)**.

Note that Equation (4) must satisfy Equation (5) because a fractal has scale invariance. In Equation (5), c is an arbitrary multiple number called the scale factor. This satisfaction means that the observed target has features of self-similarity, namely, fractality. However, the actual object cannot satisfy Equation (5) perfectly. The plots in **Figure 3(c)** have a high coefficient of determination with the slope, and the observed target follows a power law. Therefore, we can decide that the observed target has fractality by evaluating the coefficient of determination using the power law.



(a) Example of Koch curve (b) Filling with boxes (c) Calculation of fractal dimension

Figure 3 Calculation of the fractal dimension of the Koch curve by using the box-counting method

C. Convergence Estimation in Fractal Dimensional Analysis

We propose an automatic convergence estimation method using fractal dimensional analysis. This method is called convergence estimation on fractal dimensional analysis (CEFD). Here, we discuss the concept of CEFD. The next section discusses a method that considers the implementation for a computer simulation. Section V discusses an implementation method that considers an actual agent such as a robot.

The learning curve and differences in internal state variables are used to determine whether the learning reaches convergence in conventional RL. We propose the CEFD method, which is performed as follows:

1) The learning curve obtained from RL is divided into parts of arbitrary data size (*e.g.*, episodes and time).
2) The fractal dimension is calculated from each data part by the preprocessing step using fractal dimensional analysis.
3) After calculating the fractal dimension, the agent decides the convergence with an arbitrary threshold value.

Here, in process 1, the obtained learning curve does not need the full data of the learning curve.

It should be noted that in this method, we must assume that the learning curve has fractality. We already confirmed this in our previous work [12].

IV. Implementation of CEFD for Virtual Learning Agent

A. CEFD with Image Data

As mentioned above, we adopt the box-counting dimension as the method for calculating the fractal dimension. Normally, the box-counting method is used for image processing. The observed target also contains image data such as pictures and figures. Therefore, as a simple method to use CEFD for virtual learning agents, we develop CEFD by using fractal dimensional analysis with image data. This method is called CEFD*i*. **Figure 4** shows the process sequence. CEFD*i* is performed as follows:

1) The learning curve obtained from RL is divided into parts with arbitrary window size.
2) The curve at each part is fitted to a square and stored as an image data in formats such as JPEG.
3) Next, the fractal dimension is calculated from each square by the preprocessing step using fractal dimensional analysis.
4) Finally, each set of calculated fractal dimensions is plotted in a time-series graph. Then, the agent estimates the convergence using the threshold value of the fractal dimension.

We use ImageJ to calculate the fractal dimension using the box-counting dimension [17]. ImageJ is an image processing software that is distributed by the US National Institutes of Health. ImageJ has been used in various fields to calculate the fractal dimension [18]–[20].

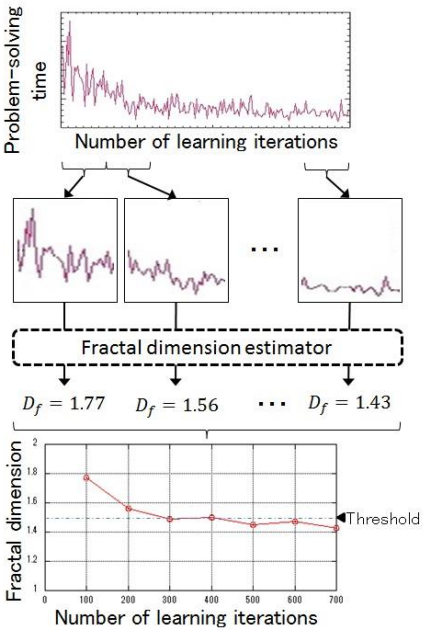


Figure 4 Simplified overview of CEFD*i*

B. *Experiment with Computer Simulation*

We conducted computer simulation experiments to confirm the effectiveness of CEFD*i* in a virtual agent that has Q-learning. We provide the following experimental conditions for the computer simulation. The goal of this experiment is to confirm the change in the fractal dimension of the learning curve that is obtained using grid worlds of four sizes. Therefore, we confirm the environmental dependency of CEFD*i*. In addition, we prove that the RL learning curve exhibits fractality through these experimental results.

Previous studies have used tasks such as zero-sum game, foraging tasks, and cooperative carrying tasks to evaluate the learning agent's performance. Here, we use a pursuit game as the evaluation function [5]. The pursuit game is a benchmark test for the agent performance, and it is measured as the time until capture. We set an $N \times N$ grid as the world. An arbitrary number of hunter agents and prey agents are deployed in this world, and we evaluate the number of steps of action until the hunters captures all of the prey. In this pursuit game, the locations for the hunters and prey are set as shown in **Figure 5**. The game ends when all of the prey has been captured by the hunters. This occurs when all hunters are adjacent to the prey at the end of a turn. After the capture of the prey, the locations of all agents are reset to their initial positions. A single episode is defined as the number of steps required to reach a state of capture. Agents act in a predefined order, such as hunter 1 $\rightarrow$ hunter 2 $\rightarrow$ prey, and one set of actions is regarded as a single step. A cell cannot be simultaneously occupied by multiple agents, and agents cannot cross the world boundaries. Moreover, hunters can learn from their own actions and perform cooperative capture actions; however, the prey cannot learn anything such as an escape action.

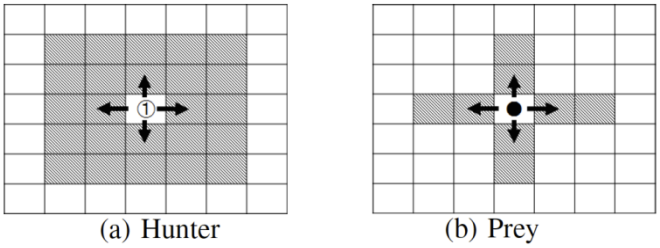


Figure 5 Hunter and prey in pursuit game. Shaded squares indicate the agents' sight range

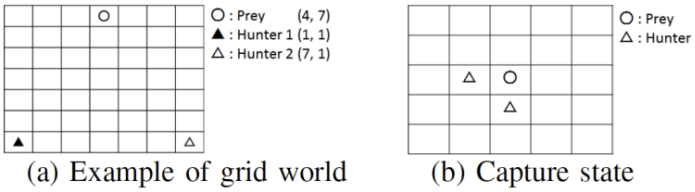


Figure 6 Pursuit game

C. *Definition of Agents*

We As mentioned above, the game involves two types of agents: hunters and prey (**Figure 6**). Only the hunters are provided with Q-learning as learning mechanisms, and the actions of the prey are defined below.

Each agent can choose one of five actions in an action space, ***A***. The agents can choose only one action per step. The action space, ***A***, is defined as

$$\boldsymbol{A} = \{\text{front}, \text{back}, \text{right}, \text{left}, \text{stop}\}. \quad (6)$$

Here, with the "stop" action, the agent keeps the same position in one step. In addition, each agent has its own sight range (indicated by the shaded cells in **Figure 6**). Initially, the hunters and prey choose their actions randomly. The hunters adjust the probabilities according to selected actions in the trial-and-error of the learning process. Although the prey does not learn, it selects an escape action when it recognizes a hunter. Note that escape actions are preprogrammed in the prey. The prey moves away from the hunter when it detects only one hunter, or in any of the possible escape directions (uniformly chosen) when it detects multiple hunters in its vicinity.

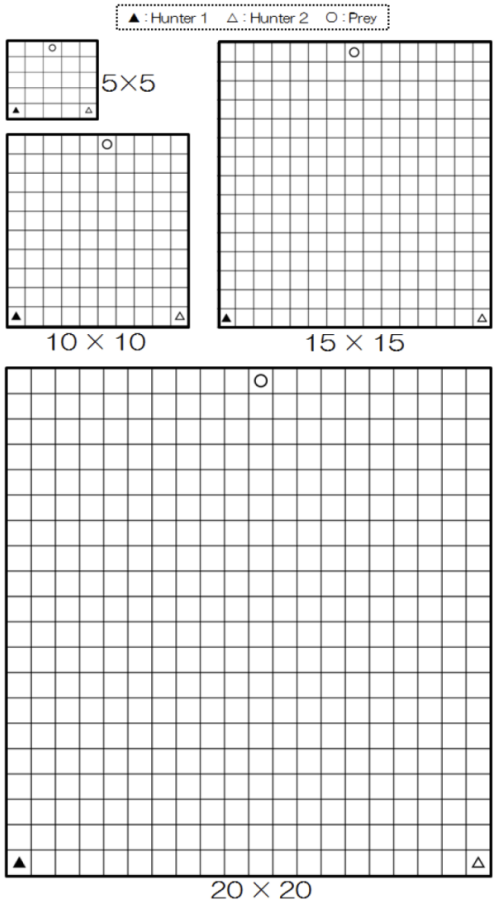


Figure 7 Four sizes of grid world

D. *Conditions*

We In the experiment, we set four grid world sizes of 5×5, 10×10, 15×15, and 20×20 (**Figure 7**). Moreover, four agent conditions are set in the four grid worlds as described below. In addition, the initial positions of agents in each experimental condition are shown in **Table I**.

TABLE I INITIAL POSITION OF EACH AGENT

Condition	Field size	Hunter 1	Hunter 2	Prey
Static	5 × 5	(1, 1)	-	(5, 5)
	10 × 10	(1, 1)	-	(10, 10)
	15 × 15	(1, 1)	-	(15, 15)
	20 × 20	(1, 1)	-	(20, 20)
Dynamic	5 × 5	(1, 1)	-	(5, 5)
	10 × 10	(1, 1)	-	(10, 10)
	15 × 15	(1, 1)	-	(15, 15)
	20 × 20	(1, 1)	-	(20, 20)
Multi-hunter	5 × 5	(1, 1)	(5, 1)	(3, 5)
	10 × 10	(1, 1)	(10, 1)	(6, 10)
	15 × 15	(1, 1)	(15, 1)	(8, 15)
	20 × 20	(1, 1)	(20, 1)	(11, 20)
Non-learner	5 × 5	(1, 1)	-	(5, 5)
	10 × 10	(1, 1)	-	(10, 10)
	15 × 15	(1, 1)	-	(15, 15)
	20 × 20	(1, 1)	-	(20, 20)

1) Static Environment: In this experimental condition, we consider that a hunter (learner) and a prey (non-learner) are deployed in the grid world, and the prey does not move. Then, the transition of the environmental states depends on the action of the hunter. This environment is MDP. This experimental condition is static compared with other environmental conditions; thus, we call this condition a *static environment*.

The observable elements s_i of the state space $\boldsymbol{S} = \{s_1, s_2, \ldots, s_n\}$ are defined as

$$s_i = (x, y). \tag{7}$$

Here, x and y are the coordinates of an agent itself. The maximum value of x and y depend on the size of the grid worlds. Note that in this experimental condition, the prey is deployed in the grid world; however, this does not affect whether the hunter can observe the prey position.

2) Dynamic Environment: The prey can move in the grid world, and a hunter is deployed in the grid world. As mentioned before, the prey can choose a random action or an escape action. This condition is different from the static condition, and the transition of the environmental states depends on not only the action of the hunter but also the action of the prey. Therefore, this environment is non-MDP, and we call this condition a *dynamic environment*.

The observable elements s_i of the state space **S** are defined as

$$s_i = (x_s, y_s, x_p, y_p). \tag{8}$$

Here, x_s and y_s are one's own coordinates. x_p and y_p are the coordinates of the prey.

3) Multi-hunter Environment: Two hunters and one prey are deployed in the grid world. The prey is allowed to move, and this environment is non-MDP. This setup is even more dynamic than that in the above environment. We call this condition a *multi-hunter environment* to differentiate it from the dynamic environment.

In this experimental setup, the observable elements s_i of the state space $\boldsymbol{S}$ are defined as

$$s_i = (x_s, y_s, x_f, y_f, x_p, y_p). \tag{9}$$

Here, x_f and y_f are the coordinates of the friendly hunter agent. This state space is the largest in this experiment.

4) Non-learner: This is essentially the same experimental condition as in the static environment. However, we modify the parameter of temperature T to 1, thus generating divergence in the learning curve. As indicated by Equation (2), any of the actions in the action space can be selected with the same probability. In other words, the hunter does not learn only in this experimental condition.

In this condition, the observable elements of the state do not affect the learning with regard to whether the hunter can observe the prey position. In this program, the state is the same as that given by Equation (7).

E. Parameters

1) Learning Parameters: The Q-learning parameters are listed in **TABLE II**. The Boltzmann parameter is changed only by the experimental conditions, and the other parameters are the same under each condition. Moreover, these parameters are the same in the four grid worlds.

TABLE II LEARNING PARAMETERS

Parameter	Value
Learning rate α	0.1
Discount rate γ	0.99
Reward r	1.0
Boltzmann parameter T	
Static	0.01
Dynamic and multi-hunter	0.005
Non-learner	1.0
Default Q-value	0.0
Number of episodes	10000
Number of trials	10

2) ImageJ Parameters: In the fractal dimensional analysis of each experiment using ImageJ, the vertical axis is adjusted to the maximum number of steps in the first 100 episodes, and the fractal dimension is calculated every 100 episodes. The parameters for the setup of ImageJ are listed in **TABLE III**.

TABLE III IMAGEJ PARAMETERS

Parameter	Value
Size of image	420 × 420 pixel
Binary of grayscale	"Autoconvert to binary"
Set scan back ground	"Let the program choose"
Positions	1
Number of sizes	50
Minimum box size	2 pixel
Maximum box size	50 pixel

F. Result

1) Results in Static Environment: The results for the change in the fractal dimension of the learning curves are shown in **Figure 8**. In **Figure 8**, each error bar in the fractal dimension indicates the standard deviation that is calculated with the change in the fractal dimension from 10 trials.

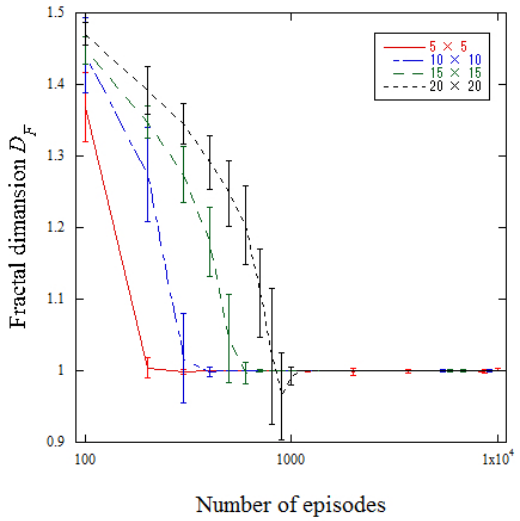


Figure 8 Result of static environment with four grid world sizes

We confirmed that the learning curves are converged with the shortest path. After the convergence of the learning curve, the change in the fractal dimension also consistently converges to 1.0, showing a clear correspondence. When the learning curve converges, the curve becomes linear. In general, the fractal dimension of the line is 1; we obtain a valid result from this experiment using CEFD*i*.

2) Results in Dynamic Environment: Under this condition, we perform an experiment in a dynamic environment. The results for the change in the fractal dimension are shown in **Figure 9(a)–(d)**.

Reaching convergence requires more time than in the static environment, and the learning curve does not converge to a single solution. In other words, the distribution of the problem-solving steps emerges from this learning curve. Moreover, the standard deviation is wider than that in the static environment's results.

3) Results in Multi-Hunter Environment: In addition, under this condition, the learning curve does not converge to a single solution as in the dynamic environment, and the converged number of steps is more likely to be higher than that of the above experimental setup. This phenomenon indicates that it is difficult for two hunters to capture the prey at the same time.

The results for the change in the fractal dimension are shown in **Figure 10(a)–(d)**. The number of steps of the learning curves is significantly different from that of the above experimental setup. However, a change in the fractal dimension is mapped from 1.0 to 2.0 dimension. Moreover, a change in the fractal dimension is more likely to converge to under 1.4 dimension, which also includes the phenomenon above by the experimental setup.

4) Results in Non-Learner: The results indicate that the change in the fractal dimension exhibits a stable transition with the four grid world sizes (**Figure 11**). Even if the hunter reiterates some episodes, the agent in this condition cannot learn.

In **Figure 11**, the fractal dimension remains at the same level between 1.4 and 1.5, the oscillation amplitude of the learning curve is different in each grid world size, and the learning curve depends on the size of the grid world.

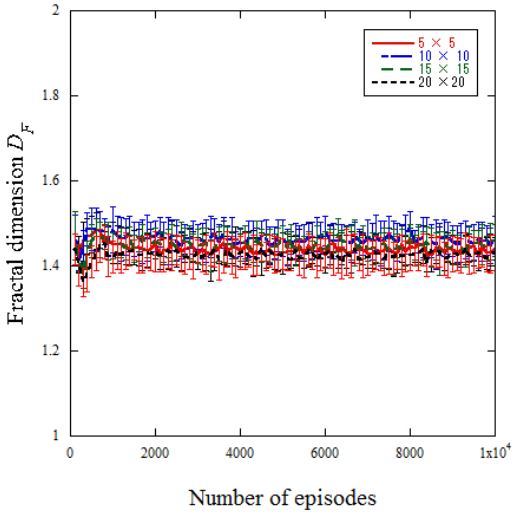


Figure 11 Result of non-learner with four grid world sizes

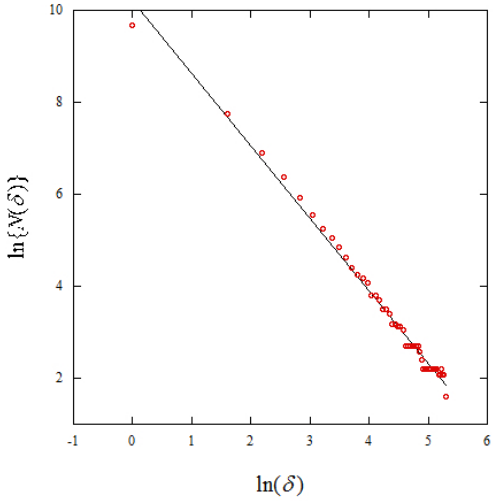
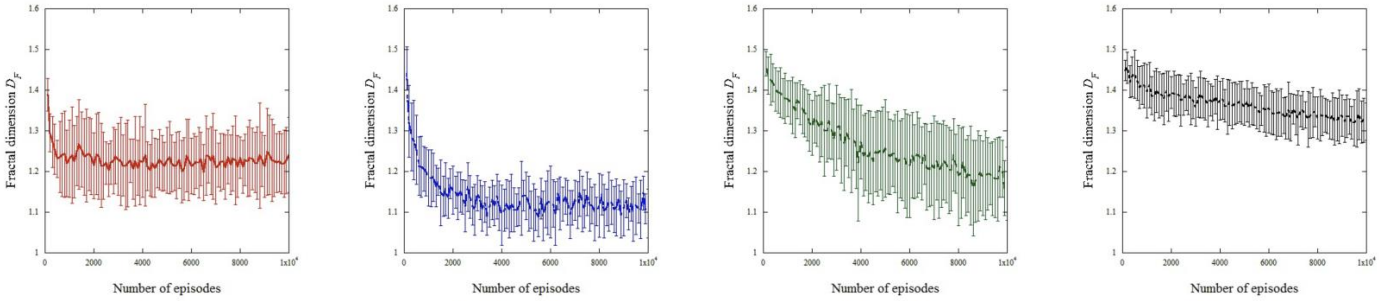
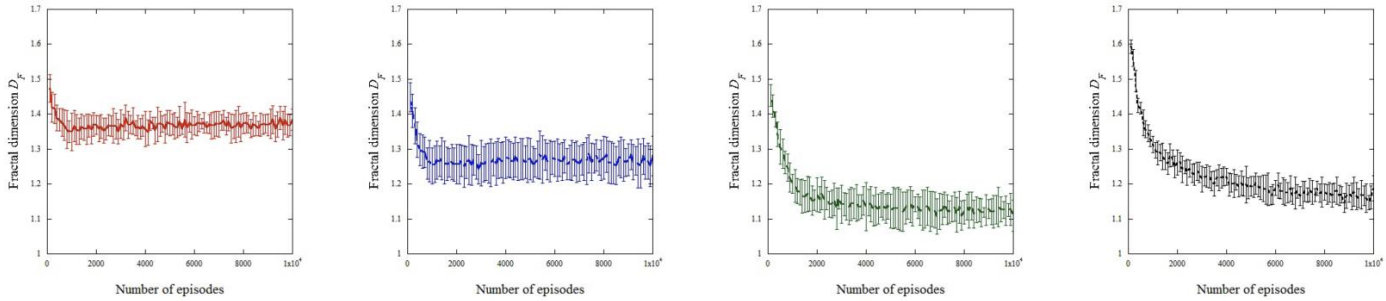


Figure 12 Instance of regression slope in multi-hunter environment with grid world size of 20×20



(a) 5 × 5 (b) 10 × 10 (c) 15 × 15 (c) 20 × 20

Figure 9 Results for dynamic environment with four grid world sizes



(a) 5 × 5 (b) 10 × 10 (c) 15 × 15 (c) 20 × 20

Figure 10 Results of multi-hunter environment with four grid world sizes

G. Discussion

1) Scale Invariance of Learning Curve: To evaluate the fractality, the observed target must have scale invariance, as mentioned in Section III-B. In the experiments, we obtained a regression slope like that shown in Figure3(c). Its determination coefficient R^2 is obtained by calculating the fractal dimension by using ImageJ. For example, one regression slope is shown in **Figure 12**; its R^2 value is 0.9926. This experimental condition is a multi-hunter environment with a grid world size of 20×20. Figure 12 shows high linearity, with a high R^2 value over 0.99. All other regression slopes also have high R^2 values over 0.9914 in all experimental conditions. Therefore, based on the box-counting dimension as the fractal dimension, it is confirmed that the learning curve has self-similarity as well as fractality.

2) Environmental Dependency of CEFDi: In relation to the above discussion, the environmental dependency of the results is discussed here. In **Figure 8**, each change in the fractal dimension is more likely to converge with the same value as the fractal dimension.

For example, in **Figure 8**, D_f converges with the same value as 1. However, the converged value of the learning curve is different in each grid world because the number of steps in the shortest path is different. This phenomenon is called scale invariance. Then, it is proven that the learning curve has fractality under the condition of calculating the fractal dimensions by the box-counting dimension.

3) Capacity for Convergence Estimation: In these experimental results, the learning curves are mapped to a fractal dimension between 1 and 2, and the shape of the change in the fractal dimension is similar in each result. This indicates the possibility that to decide the learning curve convergence, we can use a common threshold value such as 1.3~1.5. However, a fractal dimension may sometimes be under the threshold owing to measurement errors when the learning curve does not converge. Considering incorrect estimations, it is necessary to observe the change in the fractal dimension in a time series to increase the estimation accuracy.

V. Implementation of CEFD for Learning Robot

In this section, we propose the CEFD considering implementation for an actual mobile robot, and confirm the effectiveness of CEFD for an actual mobile robot with the shortest path problem as a basic experiment. The main purpose of this experiment is to test CEFD to estimate the convergence of the learning curve that is obtained from a realistic environment. In the experimental setup, the actual learning curve obtained includes a fluctuation caused by sensor noise and slip of motion. We confirm the robustness of CEFD through the experiment using an actual robot in a real environment.

A. CEFD with Numerical Data

In the previous section, we evaluated the CEFD*i* through a computer simulation. However, CEFD*i* uses image processing software to calculate the fractal dimension. The computational load of image processing becomes an issue when considering

the implementation for an actual learning agent. Therefore, to implement the CEFD for the actual learning agent, we propose a CEFD method with numerical data (CEFD*n*) and develop the algorithm of the CEFD with numerical data. In this method, the fractal dimension is computed using numerical data of the learning curve obtained from the learning agent.

In the following, we explain a novel method for calculating the fractal dimension from numerical data. The learning curve, which is the measurement target, consists of integer values that are less than or comparable to a few thousand or several tens of thousands of values. When considering the measurement of the fractal dimensions from the learning curve, this method can be improved for greater speed and simplification. The learning curve is constructed with the values of steps in a time series. The data of learning curve is described as

$$P = \{p_1, p_2, \ldots, p_n\}. \tag{10}$$

If the partial steps of the learning curve $P_i^w \subseteq P$ are defined as

$$P_i^w = \{p_i, p_{i+1}, \ldots, p_{i+w}\}, \tag{11}$$

where we assume that P_i^w is obtained by the function $\Lambda(\cdot)$ from an arbitrary *i* th episode to the *i* + *w* th episode, also *i*; *i*+*w* < *n*. Measurement interval w is the size of the window. $\Lambda(\cdot)$ is described as

$$\Lambda(P, i, w) = P_i^w. \tag{12}$$

We propose CEFD*n* expressed by Equation (13) and Equation (14) to calculate the fractal dimension approximately from the numerical data.

$$M(\delta, j) = \left\lfloor \frac{\max\{\Lambda(P_i^w, 1+\delta(j-1), \delta)\}}{\delta} \right\rfloor - \left\lfloor \frac{\min\{\Lambda(P_i^w, 1+\delta(j-1), \delta)\}}{\delta} \right\rfloor + 1, \tag{13}$$

$$N(\delta) = \sum_{j}^{w/\delta} M(\delta, j). \tag{14}$$

Here, δ is the size of box $(2 \le \delta < w \mid \delta \in \mathrm{N}^+)$. j is the number for counting $(1 \le j \le (w/\delta) \mid j \in \mathrm{N}^+)$. $M(\delta, j)$ is the number of boxes that are sufficiently filled for the learning curve using the box δ on the j th with measurement interval w. This means that $N(\delta)$ is the total number of boxes when measuring the measurement interval *w* of the learning curve using the size of the box δ. $\lfloor \cdot \rfloor$ is a floor function. Then, the fractal dimension D_f is calculated by $N(\delta)$ and δ using Equation (4).

Figure 13 shows an example of the calculation by Equations (13) and (14). In this case, for example, 10 data points of the learning curve (red lines in Figure 13) are obtained from RL. (For the actual measurement interval w episodes, for example, 100 episodes are used.) If the box size δ is set to 4, first, $M(4, 1)$ is calculated. The number of episodes and number of steps equivalent to and the number of boxes $M(4, 1)$ with $j = 1$ are shown in the following equation.

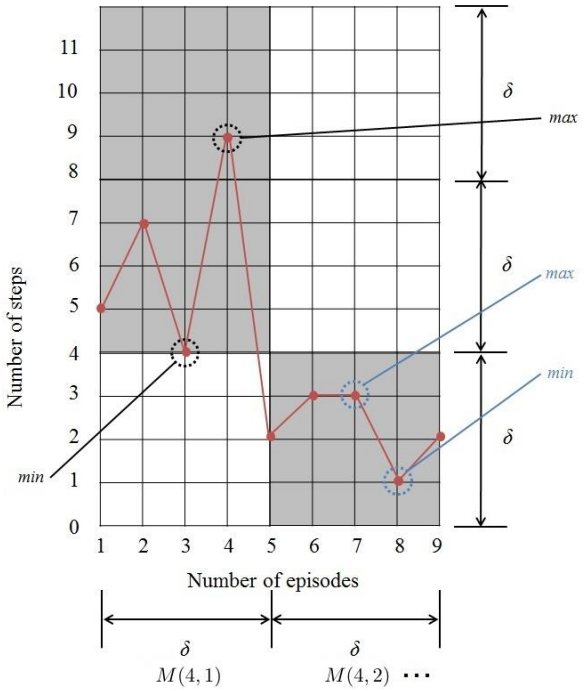


Figure 13 Example of approximated calculation for box-counting method in CEFD*n*

$$\begin{aligned} M(4,1) &= \left\lfloor \frac{9}{4} \right\rfloor - \left\lfloor \frac{4}{4} \right\rfloor + 1. \\ &= \lfloor 2.25 \rfloor - \lfloor 1 \rfloor + 1 \\ &= 2. \end{aligned} \tag{15}$$

$M(4, 1)$ is indicated by the shaded cells in Figure 13. Moreover, the number of boxes $M(4, 2)$ with $j = 2$ is

$$\begin{aligned} M(4,2) &= \left\lfloor \frac{3}{4} \right\rfloor - \left\lfloor \frac{1}{4} \right\rfloor + 1. \\ &= \lfloor 0.27 \rfloor - \lfloor 0.25 \rfloor + 1 \\ &= 1. \end{aligned} \tag{16}$$

$N(\delta)$ is calculated from the summation of the number of boxes from $M(\delta, 1)$ to $M(\delta, w/\delta)$. Moreover, the concept of the process sequence is shown in **Figure 14**. CEFD*n* differs from CEFD*i* in that the measurement window of the learning curve, that is, the measurement interval, can be moved with an arbitrary episode.

In addition, to estimate the convergence of the learning curve using CEFD*n*, a threshold value is needed as in CEFD*i*. As already mentioned in Section IV-G3, the change in the fractal dimension has some fluctuation. Therefore, considering the implementation, another index is needed except a threshold value to estimate the convergence. In this experiment, we use the sequential updating number of the lowest fractal dimension

M. The calculated fractal dimension, D_f, is below D_t, and the lowest fractal dimension is set to D'_f every time the latest lowest fractal dimension is obtained. If D'_f is not updated or D'_f is equal to the latest D_f, and this condition exceeds *M* times, the learning curve is considered to have converged.

Algorithm 1 shows the pseudocode of preformulated CEFD*n* for implementing CEFD*n* as a computer program in a learning agent. This pseudocode includes the algorithm of Q-learning.

Algorithm 1 CEFDn

Initialize: $Q(s,a)$ arbitrarily value
Input: C,D_t,w are given
Input: D_f,D'_f are given as higher than D_t
$\delta_n \leftarrow \{1, 2, \cdots, m\}$
$c \leftarrow 0$
while $c < C$ **do**

  /***Algorithm for reinforcement learning.***/
  while Goal state $\neq$ true **do**
    Observe state s
    $a \leftarrow \dfrac{\exp\left(\frac{Q(s,a)}{T}\right)}{\sum_{b \in A} \exp\left(\frac{Q(s,b)}{T}\right)}$
    Take action a
    Observe r and s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha\{r + \gamma \max_{a' \in A} Q(s',a') - Q(s,a)\}$
    $e \leftarrow e + 1$
  end while

  /***Algorithm for calculation of fractal dimension.***/
  if $e \geq w$ **then**
    $i = e - w$
    $P_i^w = \Lambda(P, i, w)$
    for all δ_n **do**
      for j to w/δ **do**
        $M(\delta_n, j) \leftarrow \left\lfloor \frac{\max\{\Lambda(P_i^w, 1+\delta_n(j-1), \delta_n)\}}{\delta_n} \right\rfloor - \left\lfloor \frac{\min\{\Lambda(P_i^w, 1+\delta_n(j-1), \delta_n)\}}{\delta_n} \right\rfloor + 1$
        $N(\delta_n) \leftarrow N(\delta_n) + M(\delta_n, j)$
      end for
    end for
    $D_f \leftarrow -\lim_{\delta \to 0} \frac{\log N(\delta)}{\log(\delta)}$

    /***Algorithm for estimation of convergence.***/
    if $D_f < D_t$ **then**
      if $D_f \geq D'_f$ **then**
        $c \leftarrow c + 1$
      else if $D_f < D'_f$ **then**
        $D'_f \leftarrow D_f$
        $c \leftarrow 0$
      end if
    else
      $c \leftarrow 0$
      Reset D'_f as default value
    end if
  end if
end while
Convergence is estimated

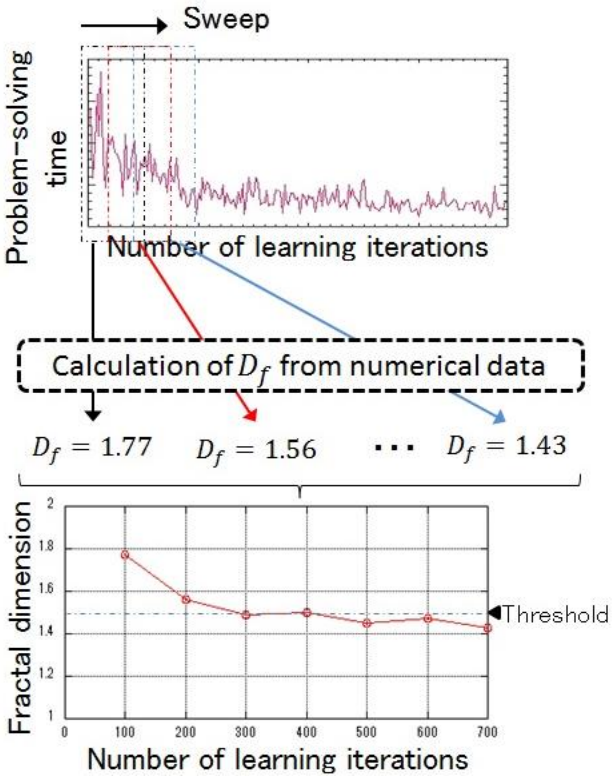


Figure 14 Simplified overview of CEFD*n*

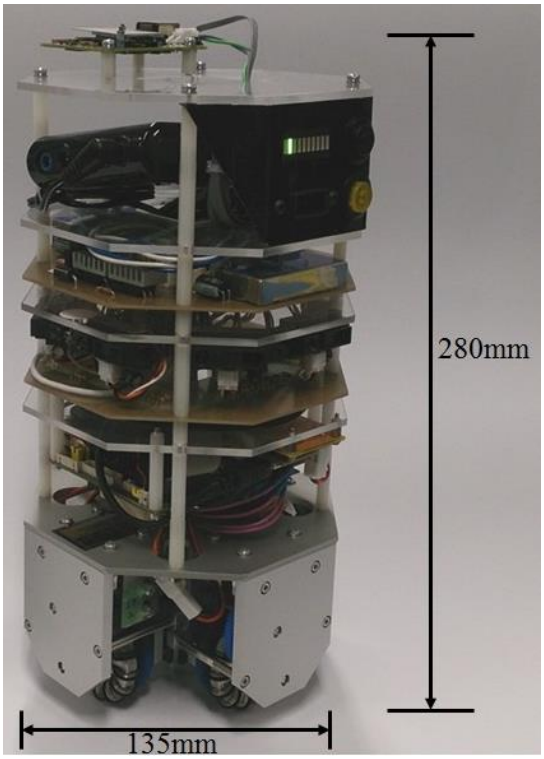


Figure 15 Autonomous omnidirectional mobile robot ZEN-Q. ZEN-Q has a four-wheel-type omnidirectional mobile mechanism and eight IR sensors. It contains an embedded computer (IA32 architecture); this computer can control the sensor and motor

B. Experiment Using Mobile Robot

In this experiment, we use a mobile robot called ZEN-Q (**Figure 15**) as an actual learning agent, and we implement CEFDn in ZEN-Q. The robot has an omnidirectional mobile mechanism and infrared (IR) sensors that are placed around the robot body in eight directions to detect the distance between the robot body and a wall. ZEN-Q can move autonomously, and the Q-learning algorithm is already implemented in it.

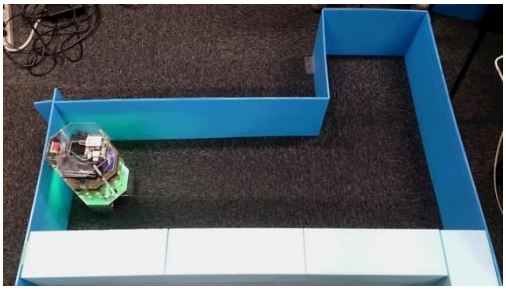


Figure 16 Photograph of the experimental environment and its initial state. In this experiment, the robot executes RL to explore the shortest path to the goal

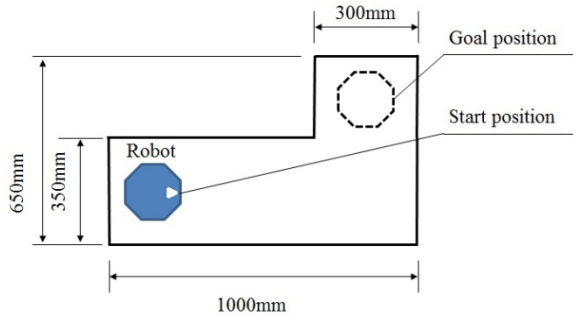


Figure 17 Simplified overview of shortest path problem with the robot. The blue octagon indicates the robot's initial position. The white octagon indicates the goal position in the shortest path problem

In this actual experiment, we adopt the task of the shortest path problem, as shown in **Figure 16**. The size of field, initial position of the robot, and goal are set as shown in **Figure 17**. The shortest path problem is also the benchmark test for agent performance, and it is measured as the time required to reach the goal position. Note that a robot is deployed in this field, which contains no movable objects. The environmental setup seems superficially MDP. However, this environment is not truly MDP because the devices of the actual mobile robot (*e.g.*, sensor and actuator) have errors including measurement errors and moving errors. This means that the transition of the environment depends on not only the decision making of the robot but also the error and slip; therefore, this experimental environment is defined as non-MDP. A single episode is defined as the number of steps required to reach a state of the goal. The start and goal positions are fixed in every experiment. When the robot reaches the goal, it is restored to the start position by a human. To judge whether the robot has reached its goal state, the robot measures the distance to the wall in three directions using three IR sensors. In this experimental setting, the robot does not rotate in the field. If the robot measures the distance in three directions but with one less than or equal to 20 cm in each three sensors, it is judged to have reached the goal.

C. Robot Specifications

As mentioned above, ZEN-Q has eight sensors and an omnidirectional moving mechanism. The robot has four movable directions, as shown in **Figure 18**. Furthermore, it has eight IR sensors, and their ranges are shown in **Figure 19**. Under normal circumstances, ZEN-Q has three degrees of freedom: x, y, and θ. However, the θ direction is made unavailable in this experiment to reduce the state space. The moving velocity is 80 mm/s in all movable directions and ZEN-Q can decide an action every 0.5 s. As an action selector, we adopted the Boltzmann distribution model given by Equation (3).

The original sensor can detect a distance longer than the range shown in **Figure 19**. Here, the measurement distances are converted to four levels. For safety purposes, the robot is preprogrammed not to select an action to move toward the wall within 0-10 cm distance.

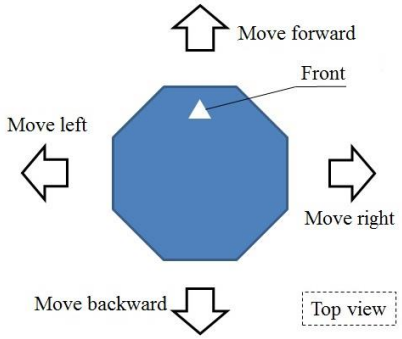


Figure 18 Top view of simplified movable directions of ZEN-Q. The robot can move with a velocity of 80 mm/s in each direction

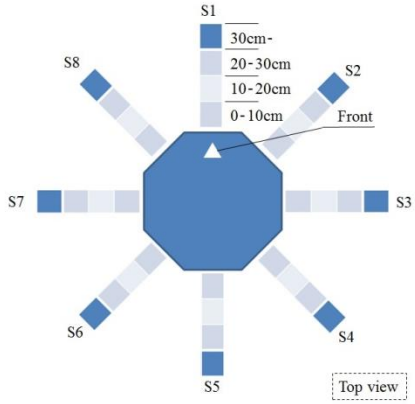


Figure 19 Top view of simplified sensor ranges of ZEN-Q. In this figure, S1–S8 indicate the eight sensors. Each sensor range has four-level resolution

D. Learning Parameters

The Q-learning parameters of ZEN-Q are shown in **Table IV**. These parameters are based on the parameters in the computer simulation, and they are preliminarily checked by the authors to learn the shortest path in this field.

The CEFD parameters are shown in **Table V**. In **Table V**, δ, p, and w are the parameters for the setup of CEFD*n*, and these parameters are also preliminarily checked. D_t and N are the parameters for the convergence estimation of the learning curve of RL.

TABLE IV LEARNING PARAMETERS OF ROBOT

Parameter	Value
Learning rate α	0.1
Discount rate γ	0.99
Reward r	1.0
Boltzmann parameter T	0.005
Default Q-value	0.0
Number of episodes	30000
Number of trials	1

TABLE V PARAMETERS OF CEFDN

Parameter	Value
Box size δ	1, 5, 10, 50, 100
Sweep interval p	1 episode
Analysis window size w (horizontal)	100 episode
Threshold value D_t	1.5
Sequential updating number C	300

E. Evaluation Index

To evaluate the accuracy of the convergence estimation, we define the estimation error rate (EER) as

$$\mathrm{EER} = \frac{\sum_{e=e_{est}-100}^{e_{est}} N_e^s - \sum_{e=e_{fns}-100}^{e_{fnl}} N_e^s}{\sum_{e=e_{fnl}-100}^{e_{fnl}} N_e^s}. \qquad (17)$$

Here, e is the episode number. e_{est} is the episode number when the convergence is estimated. e_{fnl} is the episode number when the experiment is finished; e_{fnl} is 3000. N_e^s is the number of steps when the episode number is e. If the two summed N_e^s values are different, the learning curve can decrease. This means that the learning is not finished. If the two summed N_e^s have the same value, the learning curve converges, indicating that the agent has obtained the solution.

F. Results

The results of the learning curve and change in fractal dimension are shown in **Figure 20**. The change in the fractal dimension is computed by CEFD*n* while the learning curve is obtained. The total time required for the experiment is approximately 80 h. The result shows that the learning curve exhibits a near-convergence state, and the transition of the fractal dimension also becomes a dimension of less than 1.5 as the learning curve decreases from the beginning. The longest time required to reach the goal by the actual robot is approximately 40 min, and the learning curve converges in approximately 15 s. Therefore, it can be said that the robot quickly learned to explore the shortest path from the learning curve.

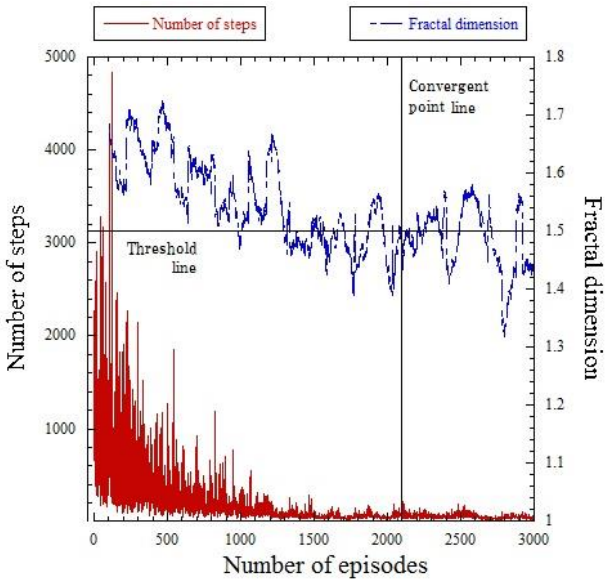


Figure 20 Learning curve and change in fractal dimension in actual robot learning

The number of episodes required for convergence is estimated by CEFD*n* to be 2101. At this time, the number of steps from the learning curve is 72.01, which is averaged from the last 100 episodes for the estimated number of episodes required for convergence. In the experiment, we performed RL with 3000 episodes. The final number of steps was 50.82 steps, averaged from the last 100 episodes.

G. Discussion

Based on the experimental results in Figure 20, EER is calculated as 43 % between the estimated number of steps for convergence and the final number of steps. This rate means that learning is not finished when CEFD*n* estimates the convergence. However, the number of steps decreases, indicating that the performance of the robot was improved from the beginning. The result of the actual robot experiment based on the shortest path problem suggests that the proposed method can estimate whether convergence will be reached.

For the change in the fractal dimension, as shown in **Figure 20**, the curve decreases but shows fluctuations. The main cause of this phenomenon is that the CEFDn algorithm depends on the shape of the learning curve. Occasionally, fluctuations such as spike noise are included in the learning curve. CEFDn is sensitive to fluctuations in the learning curve. It can be said that the CEFD*n* can estimate the convergence of the learning curve using an appropriate threshold value.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we described a method for the automatic estimation of convergence using a fractal dimensional analysis for RL. To realize knowledge co-creation when applying MARL to MARS, we proposed two methods: CEFD*i* and CEFD*n*. CEFD*i* is a method based on image processing software for calculating the fractal dimensions, and CEFD*n* is a

method based on an approximation method for calculating the fractal dimension directly from numerical data. We also conducted computer simulations and an actual environment experiment to demonstrate that the CEFD method can estimate the convergence of the learning curve obtained by RL. From the above results, we reached the following conclusions:

- The learning curve of RL exhibits fractality.
- The estimation ability does not depend on the scale of the environment.
- CEFD*i* and CEFD*n* can estimate the convergence of the learning curve based on an arbitrary threshold value.

The learner (e.g., learning agent and learning robot) can estimate the convergence of the learning curve by using CEFD with a preset threshold value that is not dependent on the scale of the environments. Therefore, it is concluded that CEFD can contribute to the application of actual learning robot systems such as MARL and our proposed knowledge co-creation. However, from the experimental results and discussion, we found the following problems in the proposed approach:

- The fractal dimension is sensitive to changes in the shape of the learning curve, such as those caused by fluctuations such as spike noise.
- In this regard, if the threshold value is set to a high fractal dimension, CEFD may provide an incorrect estimation.

In the experimental result using a robot, we set the threshold value D_t as 1.5 based on the experimental results of the computer simulation. However, despite the near of end of learning, the learning curve exceeded the 1.5 dimension many times.

In our future work, in response to the above problems, the parameters of CEFD*n* need to be adjusted. This includes the sweep interval p, analysis window size w, and threshold value D_t. Obviously, we have to compare the similarity of the change in fractal dimensions between computer simulations and actual experiments. CEFD*n* needs an evaluation based on R^2 values and the approximation error between CEFD*i* and CEFD*n*. Moreover, CEFD needs to be tested using many kinds of shapes of the learning curve considering the generalization of this method, because CEFD was tested only with a shape where the number of steps decreased from the beginning in the learning curve. We plan to demonstrate the effectiveness of CEFD*n* by conducting experiments with other types of actual robots that are not limited to robotic hardware, and environments such as toy problems.

ACKNOWLEDGEMENTS

This work was supported by the Research Institute for Science and Technology of Tokyo Denki University (Grant Number Q14J-01) in Japan, the JSPS Grant-in-Aid for Research Activity Start-Up (JP15H06102), and the JSPS Grant-in-Aid for Challenging Exploratory Research (JP16K12493). This work was funded in part by the Tough Robotics Challenge, ImPACT Program of the Council for Science, Technology and Innovation (Cabinet Office, Government of Japan), and the HAYAO NAKAYAMA Foundation for Science & Technology and Culture.

REFERENCES

[1] R. d'Andrea, "A Revolution in the Warehouse : A Retrospective on Kiva Systems and the Grand Challenges Ahead", *Automation Science and Engineering, IEEE Transaction on*, vol.9, no.4, pp.6338–639, 2012. CrossRef

[2] H.Sugiyama, T. Tsujioka, and M.Murata, "Coordination of rescue robots for real-time exploration over disaster areas", *In Proceedings of the Object Oriented Real-Time Distributed Computing (ISORC) 2008*, 11th IEEE International Symposium on, pp.170–177, 2008. CrossRef

[3] A. Marino, L. E Parker, G. Antonelli, and F. Caccavale, "A decentralized architecture for multi-robot systems based on the null-space-behavioral control with application to multi-robot border patrolling", *Journal of Intelligent & Robotic Systems*, vol.71, no.3–4, pp.423–444, 2013. CrossRef

[4] M. Tan, "Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents", *In Proceedings of the Tenth International Conference on Machine Learning*, vol.337, pp.330–337, 1993. CrossRef

[5] M. J. Matari´c, "Reinforcement learning in the multi-robot domain", Autonomous Robots, vol.4, pp.73–83, 1997. CrossRef

[6] S. Arai, K. Sycara, and T. R. Payne, "Experience-based Reinforcement Learning to Acquire Effective Behavior in a multiagent Domain", *In Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence*, pp.125–135, 2000.

[7] E. Yang and D. Gu, "A Survey on multi-agent Reinforcement Learning Towards Multi-Robot Systems", *In Proceedings of the IEEE symposium on Computational Intelligence and Games (CIG) 2005*, vol.2, 2005.

[8] M. E. Taylor, "Transfer in Reinforcement Learning Domains", vol.216, Springer, 2009. CrossRef

[9] H. Kono, A. Kamimura, K. Tomita, Y. Murata and T. Suzuki,"Transfer Learning Method Using Ontology for Heterogeneous Multi-agent Reinforcement Learning", *International Journal of Advanced Computer Science and Applications*, vol.5, no.10, pp.156–164, 2014. CrossRef

[10] M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J. Montiel, A. Perzylo, B. Schieble, M. Tenorth, O. Zwegle and R. Van de Molengraft, "A World Wide Web for Robots", *Robotics and Automation Magazine*, vol.18, no.2, pp.69–82, 2011. CrossRef

[11] G. Hu, W. P. Tay and Y. Wen, "Cloud robotics: Architecture, challenges and applications", *IEEE Network*, vol.26, no.3, pp.21–28, 2012. CrossRef

[12] H. Kono, K. Sawai and T. Suzuki, "Convergence Estimation Utilizing Fractal Dimensional Analysis for Reinforcement Learning", *In Proceedings SICE Annual Conference 2013*, pp.2752–2757, 2013.

[13] C. J. C. H. Watkins and P. Dayan, "Q-Learning", *Machine Learning*, vol.8, no.3–4, pp.279–292, 1992. CrossRef

[14] K. Sichao, H. Yamamoto and K. Yamaji, "Evaluation of co2 free electricity trading market in japan by multi-agent simulations", *Energy Policy*, vol. 38, no. 7, pp.3309–3319, 2010. CrossRef

[15] M. Otani, H. Sato, K. Hattori and K. Takadama, "Analyzing an Influence of Collecting Broken Robots in Large-scale Structure Assembly though a cooperation among Multiple Robots", *IEEJ Transactions on Electronics, Information and Systems*, vol. 133, no. 9, pp.1729–1737, 2013.(in Japanese) CrossRef

[16] B. B. Mandelbrot, "How long is the coast of Britain", *Science*, vol.156, no.3775, pp.636–638, 1967. CrossRef

[17] C. A. Schneider, W. S. Rasband and K. W. Eliceiri, "Nih image to imagej: 25 years of image analysis", *Nature methods*, vol.9, no.7, pp.671–675, 2012. CrossRef

[18] A. Forsythe, M. Nadal, N. Sheehy, C. J. Cela-Conde and M. Sawey, "Predicting beauty: Fractal dimension and visual complexity in art", *British Journal of Psychology*, Vol.102, Issue 1, pp.49–70, 2011. CrossRef

[19] F. Yas¸ar and F. Akg¨unl¨u, "Fractal dimension and lacunarity analysis of dental radiographs", *Dentomaxillofacial Radiology*, 2014. CrossRef

[20] N. N. Karle and K. M. Kolwankar, "Characterization of the Irregularity of a Terrain Using Fractal Dimension of Lakes' Boundaries", *Fractals Complex Geometry, Patterns, and Scaling in Nature and Society*, vol.23, issue 2, 2015.