Self-generation of reward by logarithmic transformation of multiple sensor evaluations

Self-Generation of Reward by Logarithmic Transformation of Multiple Sensor Evaluations

Yuya Ono[1†], Kentarou Kurashige[2],
Afiqe Anuar Bin Muhammad Nor Hakim[3] and Yuma Sakamoto[4]

[1]Division of Information and Electronic engineering, *Muroran* Institute of Technology, Hokkaido, Japan
(Tel: +81-143-46-5400; E-mail: 21043021@mmm.muroran-it.ac.jp)
[2]Department of Information and Electronic Engineering, *Muroran* Institute of Technology, Hokkaido, Japan
(Tel: +81-143-46-5400; E-mail: kentarou@muroran-it.ac.jp)
[3]Division of Information and Electronic engineering, *Muroran* Institute of Technology, Hokkaido, Japan
(Tel: +81-143-46-5400; E-mail: 20043001@mmm.muroran-it.ac.jp)
[4]Division of Information and Electronic engineering, *Muroran* Institute of Technology, Hokkaido, Japan
(Tel: +81-143-46-5400; E-mail: 22043022@mmm.muroran-it.ac.jp)

Abstract: Although the design of the reward function in reinforcement learning is important. It is difficult to design a system that can adapt to a variety of environments and tasks. Therefore, we propose a method to autonomously generate rewards from sensor values, enabling task- and environment-independent reward design. Under this approach, environmental hazards are recognized by evaluating sensor values. The evaluation used for learning is obtained by integrating all the sensor evaluations that indicate danger. Although prior studies have employed weighted averages to integrate sensor evaluations, this approach does not reflect the increased danger arising from a higher amount of more sensor evaluations indicating danger. Instead, we propose the integration of sensor evaluation using logarithmic transformation. Through a path learning experiment, the proposed method was evaluated by comparing its rewards to those gained from manual reward setting and prior approaches.

Keywords: Self-Generation of Reward, Reinforcement learning, Danger recognition

1. INTRODUCTION

Recent years have seen an increase in the socioeconomic demand for robots[1-3]. Specifically, unmanned robots have been introduced in complex and changeable environments such as disaster sites, to minimize human fatalities. Fully autonomous robots are expected to not only accomplish their tasks, but also detect hazards and avoid hardware and software failures. Therefore, reinforcement learning (RL)[4] has been used to discover appropriate environmental actions through trial and error. Owing to its versatility, RL has been employed for numerous complex and practical applications[5, 6]. Reward is an important element of RL, as the reward function's design affects the learning ability. Within learning tasks and environments, higher complexity necessitates a more thorough contextual understanding increasing the burden on the reward function's designer. Accordingly, many researches have been conducted on the design of reward functions for reducing the design burden and enabling adaptive learning. For example, there are methods such as inverse reinforcement learning which estimates rewards from actions[7], and methods that design curiosity as an internal reward[8, 9].

As an autonomous reward design method, we propose Self-Generation of Reward (SGR)[10-13], wherein rewards are generated by evaluating input values to the robot's on-board sensors. Using task- and environment-independent evaluation indicators, our objective is to help robots adapt to a variety of environments. In particular, the evaluation proposed thus far have enabled the recognition of danger.

† Yuya Ono is the presenter of this paper.

Because robots are usually equipped with multiple sensors, multiple sensor evaluations are combined to generate rewards. In the integration of sensor evaluations, only those evaluations that indicate danger are used to obtain a weighted average. However, this approach does not reflect the fact that the level of danger increases with the amount of individual sensor evaluations that indicate danger. Therefore, our integration approach is designed to vary monotonically with the evaluations. In this paper, we propose the integration of sensor evaluations using logarithmic transformation. Consequently, changes in the level of danger are reflected appropriately. Within this study, we conducted simulation experiments using path learning and compared our method's results with those obtained by two types of manual reward function, and prior approaches. Designing rewards that consider the task and environment

2. SELF-GENERATION OF REWARD

The following section, provides an outline of SGR, evaluation indicators and the calculation of sensor evaluations.

2.1. Outline of the method

The SGR model is based on the learning process of an organism in a real environment, with minimal instruction from others. Organisms act according to their evaluation of pleasure and displeasure in response to external stimuli obtained via sensory organs[14, 15]. Similarly, an agent may evaluate the current state by expressing its pleasure or displeasure based on input values picked up by on-board sensors. Furthermore, evaluations in RL can be expressed as rewards for states and actions. In other words, we presume that sensor-

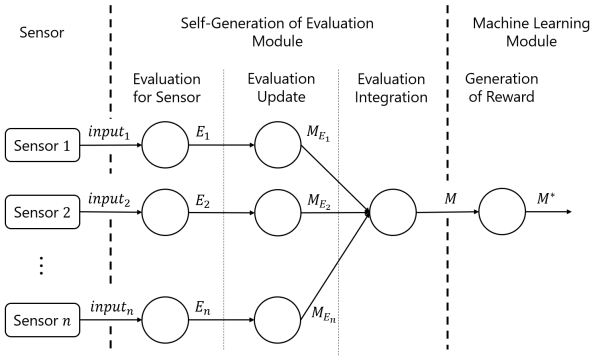

Fig. 1. System diagram of SGR

based evaluations would enable the agent to learn independently regardless of its environment or task. Figure 1 illustrates SGR system.

In SGR, environmental stimuli are acquired by the agent via sensors, and the obtained sensor values are evaluated by specific indicators to generate rewards. These indicators extend the feelings of pleasure and displeasure found in living organisms to a wide variety of sensors.

Currently, we propose three types of evaluation indicators: magnitude of input, predictability of input, and time with no input. All of these indicators are designed to target the sense of displeasure.

2.2. Evaluation Indicators

The following section describes the calculation method of the evaluation value for each indicator in SGR. These evaluation values range is from 0 to 1. The evaluation value corresponding to the magnitude of input $E_{M\ i}(t_n)$ is obtained by Eq. (1), where the constants max_i and N_i are pre-set by the sensors mounted on the robot. The magnitude of N_i determines the amount of change in the evaluation value $E_{M\ i}(t_n)$.

$$E_{M\ i}(t_n) = \left[1 + \exp\left\{\frac{\mu_i(t_n) - \frac{max_i + \delta_i(t_n)}{2}}{N_i(max_i - \delta_i(t_n))}\right\}\right]^{-1} \quad (1)$$

$E_{M\ i}$: Evaluation value for the magnitude of input,
i : Sensor number,
t_n : n-th input in action at time t,
μ_i : Average of input values,
δ_i : Threshold,
max_i : Maximum input value, N_i : Constant

We now define the average of input values $\mu_i(t_n)$ and threshold $\delta_i(t_n)$. First, the average of input values $\mu_i(t_n)$ used in the evaluation indicator is obtained by Eq. (2). This figure is used to handle multiple inputs.

$$\mu_i(t_n) = \frac{1}{n}\sum_{j=0}^{n} input_i(t_j) \quad (2)$$

μ_i : Average of the input values, $input_i$: Input value

The threshold $\delta_i(t_n)$ is the base value of evaluation, as $E_{M\ i}(t_n)$ produces a higher evaluation when the input is smaller than $\delta_i(t_n)$. $\delta_i(t_n)$ is updated for every a sensor input according to Eq. (3). This allows the agent to autonomously perform evaluations appropriate to the environment.

$$\delta_i(t_n) \leftarrow \delta_i(t_n) + \beta_i\{\mu_i(t_n) - \delta_i(t_n)\} \quad (3)$$

δ_i : Threshold, β_i : Constant

The evaluation value $E_{p\ i}(t_n)$ for predictability of input is obtained by Eq. (4) using the prediction error $D_i(t_n)$ and constant S_i. Here, $D_i(t_n)$ is calculated from the absolute value of the difference between the predicted and measured values of the input from Eq. (5). Predicted values of the input $f_i(t_n)$ are calculated using Eq. (6) by generating them using support vector regression (SVR)[16]. The input value of SVR is a constant number of inputs of sensor i from t_1 to t_{n-1}.

$$E_{P\ i}(t_n) = \exp\left\{\frac{-D_i(t_n)^2}{S_i}\right\} \quad (4)$$

$$D_i(t_n) = |input_i(t_n) - f_i(t_n)| \quad (5)$$

$$f_i(t_n) = \boldsymbol{w}^T\phi(t_n) + b \quad (6)$$

$E_{P\ i}$: Evaluation value for predictability of input,
D_i : Prediction error, S_i : Constant,
f_i : Predicted values of input using SVR,
$\boldsymbol{w}$: One-dimensional coefficient vector,
ϕ : Feature transformation functions in kernel functions,
b : Bias

The evaluation value $E_{T\ i}(t_n)$ corresponding to time with no input is obtained by Eq. (7) using the variable $d_i(t_n)$ and constant k_i. Here, a longer duration wherein the input value remains zero corresponds to a lower evaluation. The degree of reduction in evaluation is determined by the variable $d_i(t_n)$, which increments at every instance the input value is 0, according to Eq. (8).

$$E_{T\ i}(t_n) = \exp\left\{\frac{-d_i(t_n)}{k_i}\right\} \quad (7)$$

$$d_i(t_n) = \begin{cases} \gamma_i d_i(t-1_n) & (input_i(t_n) \neq 0) \\ d_i(t-1_n) + 1 & (input_i(t_n) = 0) \end{cases} \quad (8)$$

$E_{T\ i}$: Evaluation value for time with no input,
d_i : Function to detect no input, k_i : Constant,
γ_i : Attenuation

2.3. Calculation of Sensor Evaluation

The following section describe the overall calculation of sensor evaluation. Because each sensor i must be represented by a single evaluation value, the three evaluation indicators must be integrated. Accordingly, the sensor evaluation value $E_i(t_n)$ is defined as the weighted geometric mean of the values obtained from Eqs. (1), (4), and (7). The integrated value is obtained according to Eq. (9). By integrating only those evaluations lower than 0.5, which represents the midpoint of the evaluation value range, it is possible to express the danger

level as sensory information. Disregard of evaluation values is represented by setting the weights to 0 or 1 using Eq. (10). The m-squared root in Eq. (9) is determined by the sum of the weights obtained by Eq. (11). If the evaluation values for all indicators are disregarded, sensor i is disregarded completely.

$$E_i(t_n) = \begin{cases} \sqrt[m]{E_{M\,i}(t_n)^{\omega_M} \cdot E_{P\,i}(t_n)^{\omega_P} \cdot E_{T\,i}(t_n)^{\omega_T}} & (m \neq 0) \\ 0.5 & (m = 0) \end{cases} \tag{9}$$

$$\omega_x = \begin{cases} 1 & (E_{x\,i} < 0.5) \\ 0 & (E_{x\,i} \geq 0.5) \end{cases} \tag{10}$$

$$m = \omega_M + \omega_P + \omega_T \tag{11}$$

$E_i(t_n)$: Sensor evaluation value, m : Sum of weights,
ω_x : Weight of evaluation indicator x,
$E_{x\,i}$: Value of evaluation indicator x

Subsequently, the multi-input sensor evaluation value is obtained. Because multiple inputs are used simultaneously, the sensor evaluation is updated with the evaluations previous inputs along with that of the current input. The updated equation for evaluation M_{E_i} at sensor i is expressed as Eq. (12), wherein M_{E_i} is updated at each input using the discount rate γ_e, and initialized at each evaluation.

$$M_{E_i} \leftarrow M_{E_i} + \gamma_e(E_i(t_j) - M_{E_i}) \tag{12}$$

M_{E_i} : Evaluation of sensor i, γ_e : Discount rate,

Furthermore, because robots are generally mounted with multiple sensors, the sensor evaluations require integration. This process is performed using only those sensor evaluations corresponding to $m \neq 0$ in Eq. (11). First, the range of evaluation values M_{E_i} is changed from -1 to 1 using Eq. (13) for the sake of convenience. The integration equation for multiple sensor evaluations is defined as a weighted average as in Eq. (14). The weights used for this weighted average are obtained in Eq. (15).

$$M'_{E_i} = 2M_{E_i} - 1 \tag{13}$$

$$M^*_{prev.} = \frac{\sum_{i \in sensor} \omega(M'_{E_i}) M'_{E_i}}{\sum_{i \in sensor} \omega(M'_{E_i})} \tag{14}$$

$$\omega(M'_{E_i}) = \frac{\exp(-x_1(M'_{E_i} + 1))\cos(x_2(M'_{E_i} + 1) + x_3)}{x_4} \tag{15}$$

M'_{E_i} : Evaluation of sensor i after range changes,
M_{E_i} : Evaluation of sensor i,
$M^*_{prev.}$: Integrated evaluation of multiple sensors,
num : Total number of sensors to be integrated,
$\omega(M_{E_i})$: Weight coefficient,
$sensor$: Set of sensors to be integrated
$x_1,\ x_2,\ x_3,\ x_4$: constant

3. INTEGRATION OF SENSOR EVALUATIONS USING LOGARITHMIC TRANSFORMATION

This following section describes the proposed method of integrating sensor evaluations using logarithmic transformation of the sum of sensor evaluations. Although the prior study calculated the integrated evaluation by weighted average of the sum of ones, the risk detection using that integrated evaluation is problematic. The problem is that the value of only one danger is sometimes more dangerous when the evaluation of only one danger is compared to the evaluation of multiple dangers. Because the more danger there is, the more dangerous we consider the situation, it is problematic to detect dangers as described above. In this paper, we consider integrating the sensor evaluations by summing them.

However, we do not believe that the sum of the sensor evaluations and the integrated evaluation are simply proportional. This causes the greater the danger rating, the less likely one is to perceive a difference in danger. When making comparisons in integrated evaluations, how much difference depends on the ratio. Thus, there is a logarithmic relationship between the total sensor evaluation and the integrated evaluation. Therefore, in this study, the integrated evaluation is the logarithmic transformation of the sum of the sensor evaluations to express the sum of the evaluations of the situation as dangerous and the convergence to dangerous.

Equation (16) is shown as an integrated method of evaluation using a logarithmic transformation. Because the current evaluation indicators only focus on danger, a higher level of danger must correspond to a lower evaluation. The cumulative sum is therefore expressed as $1 - M_{E_i}$. The constant ρ can be adjusted according to the type and number of sensors used. The range of evaluation values is changed from Eq. (17) for use in learning.

$$M_{prop.} = 1 - \ln\bigl(1 + \rho \sum_{i \in sensor} (1 - M_{E_i})\bigr) \tag{16}$$

$$M^*_{prop.} = M_{prop.} - 1 \tag{17}$$

$M_{prop.}$: Integrated evaluation of multiple sensors,
M_{E_i} : Evaluation of sensor i, ρ : Constant,
$sensor$: Set of sensors to be integrated,
$M^*_{prop.}$: Evaluation for learning

4. SIMULATION EXPERIMENT

4.1. Experiment Summary

The experiment conducted in this study simulates a robot learning a path, with the objective of selecting a path with the minimal amount of danger, thus minimizing the potential for malfunctions. The underlying purpose of this experiment was to verify an evaluation based on the total amount of danger. The experimental setting was a 7×9 two-dimensional grid map as shown in Fig. 2. The robot moves to each Subsequent square by selecting one of four directions. The robot was able to recognize its own coordinates, and was mounted

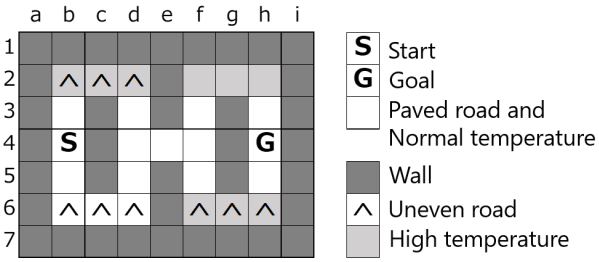

Fig. 2. Experimental environment

Table 1. Experimental settings

Number of trials	1000
Definition of one trial	End of 200 actions or Reaching the Goal
Learning Methods	Q-Learning
Action Selection Methods	ε-greedy algorithm
Goal reward	1
Reward for one action	-0.05
Learning rate α	0.3
Discount rate γ	0.99

Table 2. Agent Parameters

Move time per one square	2[s]
Sampling rate of sensors	50[inputs/s]
Maximum value of vibration sensor	10
Minimum value of vibration sensor	0
Maximum value of temperature sensor	100
Minimum value of temperature sensor	0
Maximum value of collision sensor	1
Minimum value of collision sensor	0

with three types of sensors: vibration, temperature, and collision. Four different reward designs were compared: two types of manual reward function, SGR, and the proposed method.

The experimental settings are listed in Table 1. The robot performed path learning using RL, with Q-learning as the action learning method, and ε-greedy method as the action selection method. All action values were updated according to Eq. (18). The reward r_{t+1} was determined by each of the four approaches. The random probability ε in the ε-greedy method, determined by Eq. (19), decreased with each successive trial.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_b Q(s_{t+1}, b) - Q(s_t, a_t)] \tag{18}$$

$$\varepsilon = 0.5 \times 0.99^{trial} \tag{19}$$

$Q(s_t, a_t)$: Action value, s_t : State at time t,
a_t : Action at time t, r_{t+1} : Reward,
α : Learning rate $(0 < \alpha \leq 1)$,
γ : Discount rate $(0 \leq \gamma \leq 1)$,
ε : Random probability, $trial$: Number of trials

The following parameters were used in this experiment. The agent required 2[s] to move one square, and received 50[$inputs$] for 1[s] , in accordance with Table 2. Thus, the agent received 100 [$inputs$] per action. In the first half, inputs were transmitted from the square in the current state to the sensor, and in the second half, inputs were transmitted from the square in the next state to the sensor. The input values of each sensor were set according to Eqs. (20), (21), and (22).

$$input_{vib.} = \begin{cases} 8 + 0.5\ rand & \text{(uneven road)} \\ 2 + 0.5\ rand & \text{(paved road)} \end{cases} \tag{20}$$

$$input_{temp.} = \begin{cases} 70 & \text{(high temperature)} \\ 20 & \text{(normal temperature)} \end{cases} \tag{21}$$

$$input_{coll.} = \begin{cases} 1 & \text{(collision)} \\ 0 & \text{(non-collision)} \end{cases} \tag{22}$$

$input_{vib.}$: Input value of vibration sensor,
$rand$: Random variable $(-1 \leq rand \leq 1)$,
$input_{temp.}$: Input value of temperature sensor,
$input_{coll.}$: Input value of collision sensor

4.2. Manual Reward Setting

The following section describes the manual design of the reward function in this experiment. The reward function was set for each sensor. Two types of functions were used: constant reward setting, and a reward the varies linearly with the sensor values. Constant rewards are defined by Eqs. (23)-(25).

$$r_{vib.} = \begin{cases} -1 & \text{(uneven road)} \\ 0 & \text{(paved road)} \end{cases} \tag{23}$$

$$r_{temp.} = \begin{cases} -1 & \text{(high temperature)} \\ 0 & \text{(normal temperature)} \end{cases} \tag{24}$$

$$r_{coll.} = \begin{cases} -1 & \text{(collision)} \\ 0 & \text{(no-collision)} \end{cases} \tag{25}$$

$r_{vib.}$: Reward for vibration,
$r_{temp.}$: Reward for temperature,
$r_{coll.}$: Reward for collision

Equations (26) and (27) correspond to reward functions that vary linearly with sensor input, which is randomly selected among the sensor inputs obtained from the next state's squares. However, the reward function for collisions was the same as in Eq. (25).

$$r_{vib.} = -\frac{input_{vib.} - min_{vib.}}{max_{vib.} - min_{vib.}} \tag{26}$$

$$r_{temp.} = -\frac{input_{temp.} - min_{temp.}}{max_{temp.} - min_{temp.}} \tag{27}$$

$r_{vib.}$: Reward for vibration,
$r_{temp.}$: Reward for temperature,
min : Minimum value of sensor,
max : Maximum value of sensor

The next step is to determine the reward using Eqs. (28) and (29)), wherein weights are assigned to the rewards for each sensor.

$$r_{t+1} = r_{EX} + 0.3r_{vib.} + 0.3r_{temp.} + 0.4r_{coll.} \tag{28}$$

$$r_{EX} = r_{step} + r_{goal} \tag{29}$$

r_{t+1} : Reward, r_{EX} : External reward,
r_{step} : Reward for one action, r_{goal} : Goal reward

Table 3. Parameters for evaluation indicators

	vib.	temp.	coll.
N	0.08	0.08	0.08
Initial value of δ_i	5.0	10.0	0.5
β_i	0.001	0.001	0.001
S_i	20	20	20
k_i	250	250	2×10^6

4.3. Reward Setting by SGR

This section describes how rewards are set by SGR. The parameters of various evaluation indicators are listed in Table 3. The parameters of Eq. (15) are $x_1 = 2.0$, $x_2 = 2.4$, $x_3 = 0.5$ and $x_4 = 1.1$. Reward generation using an existing method is performed according to Eq. (30), whereas reward generation using the proposed method is performed according to Eq. (31). Equation (29) is used to obtain r_{EX}. Assume a constant $\rho = 1.0$.

$$r_{t+1} = r_{EX} + M^*_{prev.} \tag{30}$$

$$r_{t+1} = r_{EX} + M^*_{prop.} \tag{31}$$

r_{t+1} : Reward, r_{EX} : External reward,

$M^*_{prev.}$: Evaluation value using existing approach,

$M^*_{prop.}$: Evaluation value using proposed method

4.4. Experimental Results and Discussion

A comparative experiment was conducted between four types of reward settings: a constant manual reward setting (Manual1), a linearly varying manual reward setting (Manual2), an existing reward generation method (PrevSGR) and the proposed method (PropSGR). Tables 4 and 5 list the reward values for the up and down actions at coordinates (3, b) and (5, b) respectively, as well as theup and down actions at coordinates (3, f) and (5, f) respectively, which represent the branching points in the path learning process. Comparing the (3, b) and (5, b) routes, the (5, b) route is preferable, as it exhibits less danger than the (3, b) route. Likewise, the (3, f) route is preferable in the latter pair. From Tables 4 and 5, Manual1, Manual2, PropSGR are given the reward to select the ideal path. However, in PrevSGR, the reward for the (5, b) route is lower than that for the (3, b) route. Table 6 lists each sensor's evaluations as details of the reward obtained by PrevSGR. The evaluation value of 0.5 was not used for reward generation. Table 6 shows that the (5, b) route includes only one dangerous sensor evaluation, whereas the (3, b) route has two such evaluations. Because both sensor evaluations for (3, b) exceed that for (5, b), averaging will not result in a lower evaluation for the former. Consequently, the (3, b) route was deemed more rewarding in PrevSGR. Likewise, the reward for the (5, f) route was higher than that for the (3, f) route even after averaging the evaluations. Details corresponding to the rewards obtained by the proposed method are listed in Table 7. In particular, a comparison between the (3, b) and (5, b) routes reveals a lower evaluation corresponding to multiple sensor evaluations of danger.

Table 4. Reward comparison of manual settings

(coordinate, action)	Manual1	Manual2
((3, b), up)	-0.65	-0.500796
((5, b), down)	-0.35	-0.341027
((3, f), up)	-0.35	-0.307131
((5, f), down)	-0.65	-0.509011

Table 5. Reward comparison of SGR settings

(coordinate, action)	PrevSGR	PropSGR
((3, b), up)	-0.756178	-1.017360
((5, b), down)	-0.874597	-0.688434
((3, f), up)	-0.639627	-0.635723
((5, f), down)	-0.824031	-0.993790

The numbers of actions in each trial for each reward design are displayed in Fig. 3. These results demonstrate that learning converges for all methods. Furthermore, the timing of convergence is slightly faster for manual reward designs. However, the purpose of SGR is to reduce the burden of reward design, so we believe that a slight error is not a problem for the performance difference. We therefore conclude that SGR achieves equivalent performance to that of manual reward setting.

The learning paths in each reward design are shown in Fig. 4. These results indicate that only PrevSGR deviated from the ideal path, whereas all other methods learned the ideal path. Based on the previous discussion, it is considered that the ideal path could not be learned because an appropriate reward was not given in PrevSGR. PropSGR was able to learn the ideal path as efficiently as the manually-set reward function. Therefore, we were able to perform RL without directly designing a reward function.

5. CONCLUSION

This study focused on the issue of misleading danger evaluation under the conventional approach. To mitigate this issue, a monotonically varying integration was performed according to evaluation values. Furthermore, a logarithmic transformation was employed to appropriately integrate the

Table 6. Rewards and sensor evaluations in PrevSGR

(coordinate, action)	Reward	Sensor evaluation		
		vib.	temp.	coll.
((3, b), up)	-0.756178	0.115371	0.196597	0.5
((5, b), down)	-0.874597	0.0877015	0.5	0.5
((3, f), up)	-0.639627	0.5	0.205187	0.5
((5, f), down)	-0.824031	0.0665316	0.218741	0.5

Table 7. Rewards and sensor evaluations in PropSGR

(coordinate, action)	Reward	Sensor evaluation		
		vib.	temp.	coll.
((3, b), up)	-1.017360	0.155395	0.213625	0.5
((5, b), down)	-0.688434	0.106486	0.5	0.5
((3, f), up)	-0.635723	0.5	0.203711	0.5
((5, f), down)	-0.993790	0.129237	0.301062	0.5

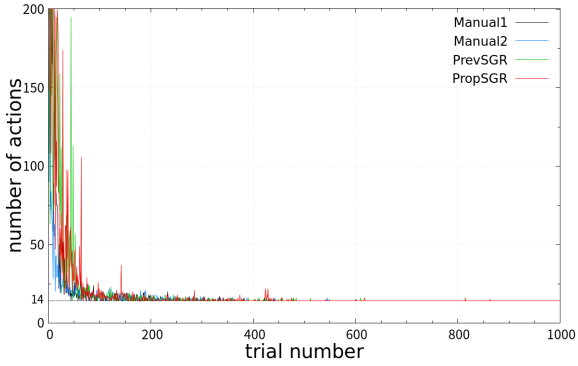

Fig. 3. Number of actions per trial

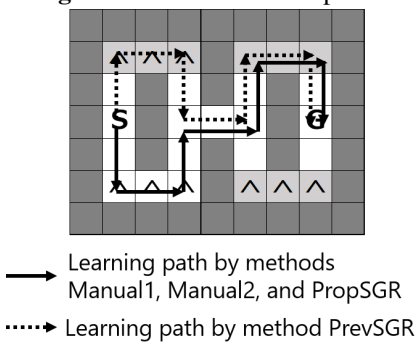

Fig. 4. Learning pass

values. Experiments also confirmed that rewards were accurately obtained in response to multiple sensor evaluations indicating danger.

As future work we have something to work on. It is an experiment in an environment close to the actual environment. In this experiment, sensor values and the environment were set up as simulations, so they different from the actual values. Therefore, to achieve the purpose of adaptation to the actual environment, it is necessary to do it with actual sensors and in the actual environment. Many parameters are set manually according to the sensor, so automatic setting or clarification of setting guidelines is needed.

REFERENCES

[1] Tetsushi Kamegawa and Taichi Akiyama and Satoshi Sakai and Kento Fujii and Kazushi Une and Eitou Ou and Yuto Matsumura and Toru Kishutani and Eiji Nose and Yusuke Yoshizaki and others, "Development of a separable search-and-rescue robot composed of a mobile robot and a snake robot", *Advanced Robotics*, Vol. 334, No. 2, pp. 132–139, 2020.

[2] Dheeraj Salgotra and Abul Hasan Khan and Hussain Mithaiwala and Murtaza Mithaiwala and Roopa B. Kakkeri, "Restaurant Waiter Robot", *SAMRIDDHI: A Journal of Physical Sciences, Engineering and Technology*, Vol. 12, No. SUP 2, pp. 46–49, 2020.

[3] Jiro Shimaya and Yuichiro Yoshikawa and Hirokazu Kumazaki and Yoshio Matsumoto and Masutomo Miyao and Hiroshi Ishiguro, "Communication support via a tele-operated robot for easier talking: case/laboratory study of individuals with/without autism spectrum disorder", *International Journal of Social Robotics*, Vol. 11, No. 1, pp. 171–184, 2019.

[4] Richard S. Sutton, Andrew G. Barto, "ReinforcementLearning, Second Edition", *The MIT Press*, 2018.

[5] S. Moazami and P. Doerschuk, "Modeling Survival in model-based Reinforcement Learning", *2020 Second International Conference on Transdisciplinary AI (TransAI)*, pp. 17–24, 2020.

[6] Lukasz Kaiser and Mohammad Babaeizadeh and Piotr Milos and Blazej Osinski and Roy H. Campbell and Konrad Czechowski and Dumitru Erhan and Chelsea Finn and Piotr Kozakowski and Sergey Levine and others, "Model-based reinforcement learning for atari", *arXiv preprint arXiv:1903.00374*, 2019.

[7] Saurabh Arora and Prashant Doshi, "A survey of inverse reinforcement learning: Challenges, methods and progress", *Artificial Intelligence*, Vol. 297, pp.103500, 2021.

[8] Mirco Mutti and Marcello Restelli, "An intrinsically-motivated approach for learning highly exploring and fast mixing policies", *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34, No. 4, pp.5232–5239, 2020.

[9] Jinxin Liu and Donglin Wang and Qiangxing Tian and Zhengyu Chen, "Learn goal-conditioned policy with intrinsic motivation for deep reinforcement learning", *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36, No. 7, pp. 7558–7566, 2022.

[10] Kentarou KURASHIGE and Kaoru NIKAIDO, "Self-Generation of Reward by Moderate-Based Index for Senor Inputs", *Journal of robotics and mechatronics*, Vol. 27, No. 1, pp. 57–63, 2015.

[11] Masaya Ishizuka and Kentarou Kurashige, "Self-Generation of Reward by Inputs from Multi Sensors-Integration of Evaluations for Inputs to Avoid Danger", *2018 International Symposium on Micro-NanoMechatronics and Human Science (MHS)*, pp. 1–7, 2018.

[12] Afiqe Anuar bin Muhammad Nor Hakim and Koudai Fukuzawa and Kentarou Kurashige, "Proposal of Time-based evaluation for Universal Sensor Evaluation Index in Self-generation of Reward", *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1161–1166, 2020.

[13] Yuya Ono and Kentaro Kurashige and Afiqe Anuar Bin Muhammad Nor Hakim and Sosuke Kondo and Kodai Fukuzawa, "Proposal of Self-generation of Reward for danger avoidance by disregarding specific situations", *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–6, 2021.

[14] Moe Watanabe and Minoru Narita, "Brain reward circuit and pain", *Advances in Pain Research: Mechanisms and Modulation of Chronic Pain*, pp. 201–210, 2018.

[15] Celina A. Salcido and Maxine K. Geltmeier and Perry N. Fuchs, "Pain and decision-making: interrelated through homeostasis", *The Open Pain Journal*, Vol. 11, No. 1, pp. 31–40, 2018.

[16] Fan Zhang and Lauren J. O'Donnell, "Support vector regression", *Machine Learning*, pp. 123–140, 2020.