Recovery Motion Learning for Arm Mounted Mobile Crawler Robot in Drive System's Failure [⋆]

Tasuku ITO [*] Hitoshi KONO [*] Yusuke TAMURA [*] Atsushi YAMASHITA [*] Hajime ASAMA [*]

[*] *Department of Precision Engineering, Graduate School of Engineering, The University of Tokyo*
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan (e-mail: {ito,kono,tamura,yamashita,asama}@robot.t.u-tokyo.ac.jp)

Abstract: In the disaster area, an arm mounted crawler robot is leveraged for missions such as searching victims. However, the robot system has possibility of failure of drive system at the extreme environment. Moreover, the robot needs to keep moving to repair the mechanism, if the drive system becomes failure. In response to this problem, it is important to realize the recovery motion. However, designing of recovery motion is difficult because the recovery motion depends on the environments and configurations of the robot. This paper describes the learning methodology of the recovery motion in the single-arm mounted crawler robot, and we confirmed that the proposed system can learn the recovery motion in computer simulation.

Keywords: Reinforcement learning, Fault tolerance, Crawler robot, Teleoperation, Motion control, Intelligent driver aids, Normalized energy stability margin

1. INTRODUCTION

A variety of mobile robots have been developed in the recent years and they have been achieving a large number of accomplishments at disaster sites (Murphy [2004], Matsuno and Tadokoro [2004]). As a matter of fact, a large number of arm mounted mobile crawler robots were implemented to disaster sites arising from the Great East Japan Earthquake that occurred in 2011. It should be noted that a feature of such robots is that they have an arm mounted for removing debris at disaster sites, as cited by referenced paper (Strickland [2014]).

Environments where these robots are implemented aredangerous areas and preparations for unforeseen accidents must be in place. This research therefore considered situations where a failure occurs with such implemented robots. In instances where a robot malfunctions, it must be returned to a location where it does not get in the way of other robots with their work operations and where it can be repaired. While a robot can continue to move if a component other than the crawlers fail, it would be difficult for a robot to move with a failed crawler. The utilization of remaining functions (such as that of the arm) can be considered as a useful ways of mobility for a robot in a crawler system failure. Such a way is called as the recovery motion in this paper.

Failures of a crawler may be due to a variety of reasons, including a failure of a wheel, motor or damages sustained by one side of crawler. The environment in which such robots operate, furthermore, include such environments as inclines or uneven road surfaces. It is therefore difficult to design a recovery motion that can be applied to all such situations. There are also a variety of types of robots and it is important to take into consideration the application of recovery motions to many robots, such as the ASTACO-SoRa with multiple arms (Strickland [2014]). Although it would be important to create redundant and predicted motions that respond to all elements described above for the future. However, it would be difficult to design all conceivable motions. It would therefore be more effective for the system to learn recovery motions in advance through the 3D simulator, rather than having humans design such motions.

The study on the method for the acquisition of recovery motion in cases of failures using reinforcement learning for a multiple legged robot as well as the study on the acquisition of walking motion by a multiped robot using the central pattern generator (CPG) can be cited as relevant studies (Kober and Peters [2012]). The acquisition of motion (Ito et al. [2003]) by a multiped robot using a reinforcement learning, furthermore, involved a method that combined the use of a genetic algorithm and motion acquisition continues when a failure occurs. It is evident from the literatures described above that the reinforcement learning is useful for the acquisition of motions of robots. There were cases that considered tumbling of robots. It is essential that the stability of robots while recovery motion should be taken into consideration when dealing with recovery motions of crawler robots. Stability will therefore be taken into consideration, along with the use of means other than the crawlers, such as the arm, for utilization in the movement of a robot.

[⋆] This work was in part funded by ImPACT Program of Council for Science, Technology and Innovation (Cabinet Office, Government of Japan).

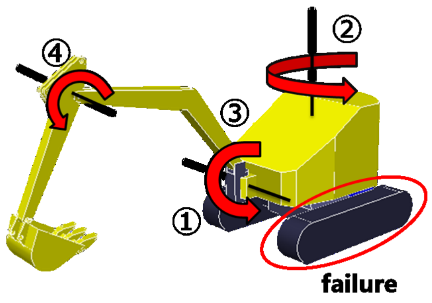

Fig. 1. Single-armed crawler robot

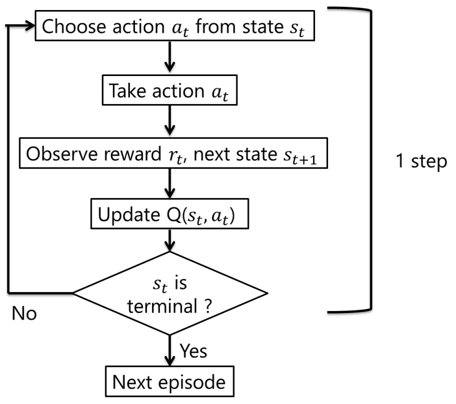

Fig. 2. Q-learning flow

The crawler robot of this research acquired recovery motion for robot in crawler system failure through a reinforcement learning. A method for recovery motion of a single armed crawler robot that is equipped with two crawlers and a single arm (Fig. 1), involving a straight movement by the robot on a flat ground, is proposed as a basic consideration of this research.

2. PROPOSED METHOD

2.1 Assumptions

This paper describes a crawler-type robot that is mounted with a single arm, as described in Fig. 1. Let us assume a situation where the robot is unable to move due to a failure with one of the crawlers. The remaining mechanisms that control the robot consists of four components, namely the crawler, swing, boom and arm (Fig. 1).

2.2 Reinforced learning

Reinforced learning is a framework of learning optimized actions acquired by a robot, through a repetition of trial and error (Sutton and Gbarto [1998]). When the robot takes an action stochastically and if such an action turns out to be an action that is the intended purpose of the action, a scalar quantity referred to as the reward is gained. The robot gradually learns to take actions that maximize the reward by repetation of taking actions. In other words, a robot can be made to learn actions that are suitable for the environment by providing rewards according to the targeted objectives of the robots through reinforcement learning. The Q-learning algorithm, which is used for many researches in reinforcement learning process and the selecting of actions by Boltzmann distribution, that makes it easier for a robot to make selections with greater rewards as the number of learning sessions increases, were used for the purpose of this research. The Q-learning is described below.

2.3 Q-learning

Q-learning is a reinforcement learning method that involves repeated selection of actions and updating of value functions during a single episode which is the term from the first state to the terminal state (Fig. 2). The updating of the action value function $Q(s_t, a_t)$ is performed using the temporal difference (TD) error δ as shown by (1), as shown by (2). $Q(s_t, a_t)$ is a function that expresses the value when action a_t is taken at state s_t, while r_t is the reward from taking such an action. $\max_a Q(s_{t+1}, a)$ is the maximum action value function Q at state s_{t+1}. The span from the selection of the action to the updating of the value function is referred to as one step and this is repeated until an episode is completed. The process moves onto the next episode when an episode has been completed.

$$\delta = r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \tag{1}$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\delta \tag{2}$$

where α is the learning rate and γ is the discount rate.

2.4 Boltzman distribution

The selecting of actions by Boltzmann distribution is a selection method for actions specified by (3) shown below, which indicates the probability $\pi(s, a)$ of action a being taken at state s $(s \in S, a \in A)$.

$$\pi(s, a) = \frac{\exp(\frac{Q(s,a)}{T})}{\sum_{b\in A} \exp(\frac{Q(s,b)}{T})} \tag{3}$$

When the temperature coefficient T is extremely large, the contribution of the action value function Q becomes small and all actions approach equivalent probability. When T is small, on the other hand, the contribution of Q becomes greater and the action probability becomes more susceptible to the influence from the magnitude of Q. The following stipulation was defined for T in this instance, using a function that approaches 0 with the progression of an episode.

$$T = \frac{1}{\log(t + 0.1)} \tag{4}$$

where t is the number of episodes.

2.5 Rewards functions

The rewards design is one of the most important designs for reinforcement learning. A large amount of time is considered to be required for the convergence of learning, since the environment is spacious and there is a large degree of freedom for the robot. Typically, in reinforcement learning the robot can obtain the reward when it arrive at the target or achieve some task. The straight movement of the robot involves the use of relocation vectors as well as the arrival at the targeted destination as rewards.

The potential tumbling of the robot was also taken into consideration for the purpose of this research, as described earlier. The reward for the stability margin of the robot was designed in addition to both the relocation vectors and the goal to promote learning, as described below. Respective rewards are described below.

Relocation vectors The angle θ ($-\pi \leq \theta \leq \pi$), formed by the unit vector $\boldsymbol{e}$ in the direction in which the robot is facing and the vector $\boldsymbol{p}$ that indicates actual progression, is defined as shown in Fig. 3. The projection of $\boldsymbol{p}$ with respect to $\boldsymbol{e}$ shall be d_1, where the direction of vector $\boldsymbol{e}$ shall be in the positive direction. d_2 shall also be defined, based on the following equation.

$$\boldsymbol{p} = \begin{bmatrix} x_{t+1} - x_t \\ y_{t+1} - y_t \end{bmatrix} \tag{5}$$

$$d_1 = |\boldsymbol{p}| \cos\theta_t \tag{6}$$
$$d_2 = |\boldsymbol{p}| \sin\theta_t \tag{7}$$

The reward $r_t^{(1)}$ is given for each single step taken, based on such values and according to the equation described below.

$$r_t^{(1)} = \eta d_1 - \lambda |d_2| - \tau |\phi| \tag{8}$$

where η and λ shall respectively be positive coefficients. The relocation vector in the straightmovement direction is evaluated based on the first term on the right side, while the second term is used to evaluate the drifting of the motion from the direction of the straight movement. The third term, furthermore, is used to evaluate the drifting of the orientation of the robot.

Goal A goal and a goal line was specified ahead of the robot at the distance b meters (Fig. 4). The segment leading to the arrival of the robot at the goal line is considered a single episode and the robot is given a reward upon reaching the goal line, based on the following equation. d_3 is the distance between the robot and the goal at the time the robot reaches the goal line.

$$r_t^{(2)} = \begin{cases} \zeta \exp(-|d_3|) & \text{if robot reached goal line} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Here ζ , furthermore, is a positive coefficient. The framework was set in such a way that the amount of reward decreases as the robot veers away from the direction of the straight movement. The integrated movements using an arm, conducted until the robot reached the goal, could be evaluated by setting details described above. The segment leading to the arrival of the robot at the goal line was considered a single episode. As soon as the robot reached the goal line, another goal line was set up ahead of the robot at the point where it has reached the current goal line, at distance b meters away. The next episode started then.

2.6 Stability evaluation

There is a potential for the robot to tumble. An evaluation on the tumbling stability of the robot was incorporated into the reward function for this reason. The normalized

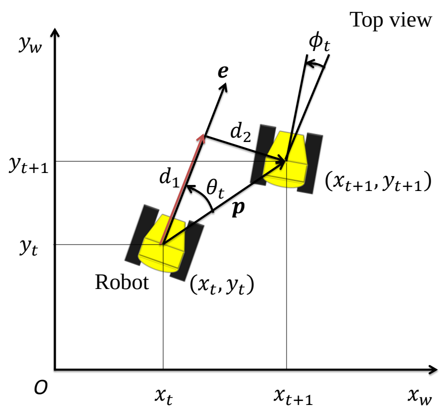

Fig. 3. Vector reward

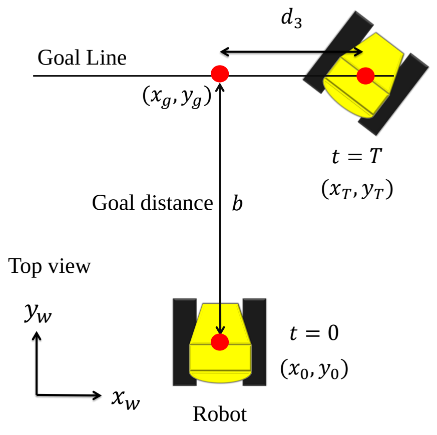

Fig. 4. Goal reward

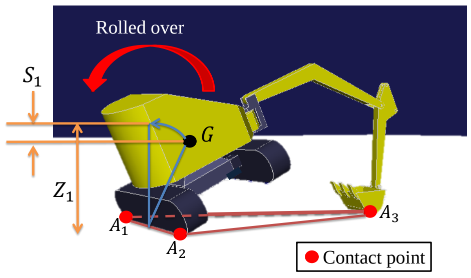

Fig. 5. Normalized energy stability margin

energy stability margin (NE stability margin) (Messuri and Klein [1985]) (Hirose et al. [2001]) was used for the purpose of stability evaluation. This is an evaluation method that should be used on the walking machines but the front and rear sections of the crawler portions were respectively assumed as support legs in this research. The height of the center of gravity of the robot was set to z_g and the maximum point of the center of gravity at the time the robot tumbles around A_1A_2 was set to Z_1 (Fig. 5). After the calculation of the height of each tumbles around A_2A_3, A_3A_1, S_i is given by following equation.

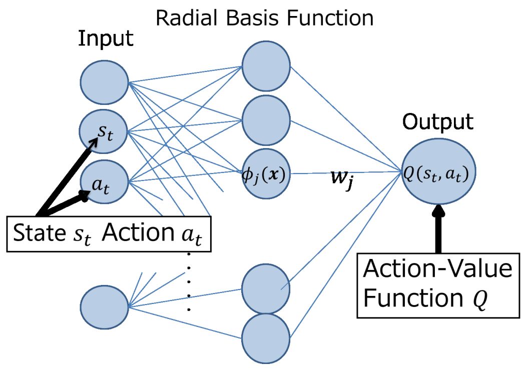

Fig. 6. RBF network

Table 1. State s_t, Action a_t

Joint	**Fig. 1**	**State** s_t	**Action** a_t
Right crawler	1	Torque (N · m)	0.3, 0, -0.3 m/s
Swing	2	Joint angle (rad)	0.3, 0, -0.3 rad/s
Boom	3	Joint angle (rad)	0.3, 0, -0.3 rad/s
Arm	4	Joint angle (rad)	0.3, 0, -0.3 rad/s

$$S_i = Z_i - z_g \tag{10}$$

The stability margin S_{NE} is given by the following equation.

$$S_{NE} = \min_i S_i \tag{11}$$

The stability is greater as S_{NE} is greater. The reward was set as a negative reward since this was handled by the reward function.

$$r_t^{(3)} = -\rho \exp(-S_{NE}) \tag{12}$$

A reward of the NE stability margin was given for each step taken. A significant additional negative reward was given when the robot tumbled, after which it was returned to the initial position and then the episode moved onto the next episode.

2.7 Function approximation of the policy

There are instances where an explosion of state space occurs with learning, where the state action space increases exponentially from state space. The radial basis function (RBF) network (Platt [1991]) (Fig. 6) was therefore used to approximate the action value function Q. The state value and action value were given as input values and the equation for the output layers is shown below.

$$y = \sum_j w_j \phi_j(\boldsymbol{x}) \tag{13}$$

The RBF network is a simple three-layer neural network, using the Gaussian function of the RBF expressed by the following equation as activation function.

$$\phi_j(\boldsymbol{x}) = \exp\left[-\sum_i \frac{(x_i - \mu_{ij})^2}{\sigma_j^2}\right] \tag{14}$$

where μ_{ij} and σ_j respectively represent center position and standard deviation of the j center layer unit.

Table 2. Experiment setup parameter

Parameter		**Value**
Learning rate	α	0.1
Discount rate	γ	0.9
Vector reward	η	8.0
Vector reward	λ	2.0
Vector reward	τ	10.0
Goal reward	ζ	10.0
Goal distance	b	2.0
Rolled over	ρ	1.0
RBF update rate	α_w	0.1
RBF update rate	α_σ	0.01
RBF update rate	α_μ	0.1

The updating of the parameters of the RBF network is described next. The value of the output value $y = Q(s_t, a_t)$ after an update is considered $\hat{y}$ and the square error E was defined as expressed by (15).

$$E = \frac{1}{2}(\hat{y} - y)^2 \tag{15}$$

The weights of w_j and μ_{ij} were updated for each updating of the Q value using the following updating equation, in order to minimize the error of output of the RBF network. α_w and α_μ were updating rates.

$$w_j \leftarrow w_j - \alpha_w \frac{\partial E}{\partial w_j} \tag{16}$$

$$\sigma_j \leftarrow \sigma_j - \alpha_\sigma \frac{\partial E}{\partial \sigma_j} \tag{17}$$

$$\mu_{ij} \leftarrow \mu_{ij} - \alpha_\mu \frac{\partial E}{\partial \mu_{ij}} \tag{18}$$

These updating equations are expressed in the following manner, using the TD error δ used for updating.

$$w_j \leftarrow w_j + \alpha_w \alpha^2 \delta \phi_j(\boldsymbol{x}) \tag{19}$$

$$\sigma_j \leftarrow \sigma_j + \alpha_\sigma \alpha^2 \delta w_j \frac{||x_i - \mu_{ij}||^2}{\sigma_j^3} \phi_j(\boldsymbol{x}) \tag{20}$$

$$\mu_{ij} \leftarrow \mu_{ij} + \alpha_\mu \alpha^2 \delta w_j \frac{x_i - \mu_{ij}}{\sigma_j^2} \phi_j(\boldsymbol{x}) \tag{21}$$

3. COMPUTER SIMULATION

3.1 Simulator

The integrated GUI software for robots, the Choreonoid (Nakaoka [2012]), was used as the dynamics simulator. The Choreonoid is equipped with a 3D dynamic simulation engine. The operating environment of the computer simulator was Ubuntu 14.04 LTS and Intel Core i7-4720 HQ, 2.6 GHz.

3.2 Experiment conditions

The robot was 1.6 meters long, 1.25 meters wide and 1.36 meters high, excluding the arm. The inputs of the status and actions for this instance were specified as described in Table 1. A total of eight values for state s and action a with relation to the four types of control targets, as well as the two values of roll and pitch of the robot for the given

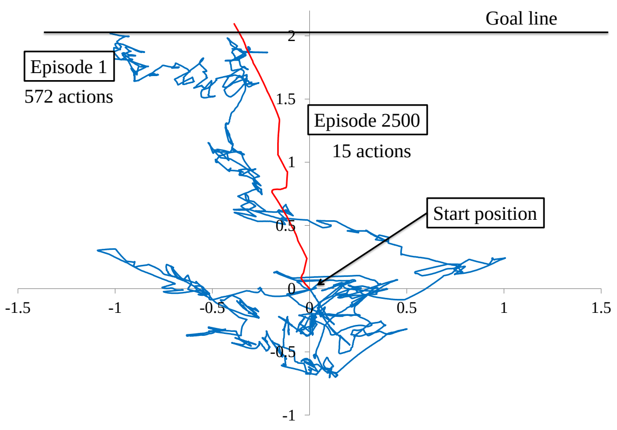

Fig. 7. Robot's trajectory

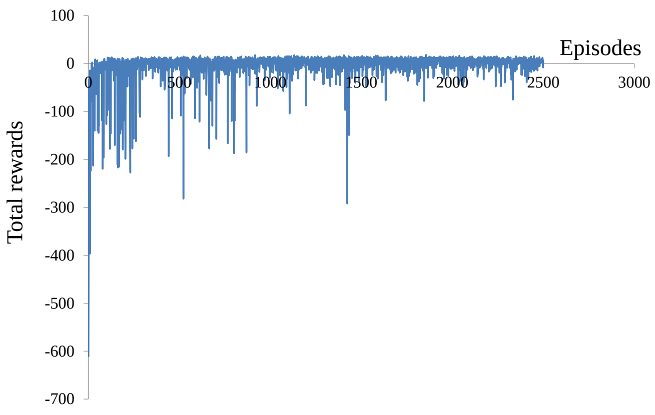

Fig. 8. Total reward

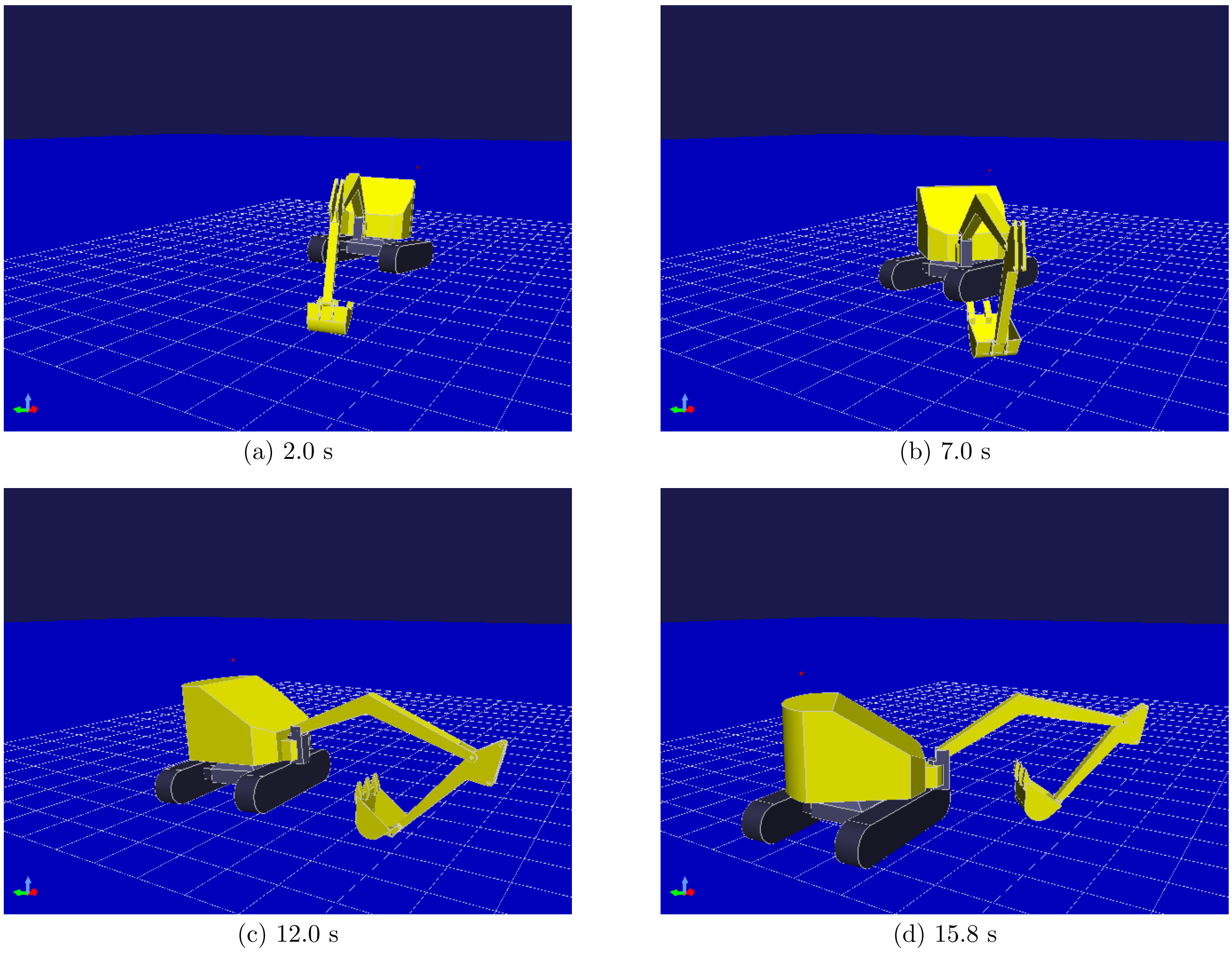

Fig. 9. Recovery motion of straight movement

state s were combined for a total of 10 values, which were normalized and then entered as input values to output the Q value through the function approximation. The output torque speed was achieved with respect to the action value a by PID control, as shown in the table. The 50 basis functions ϕ were set for the RBF. The settings for the parameters of the reinforcement learning, as well as the parameters of the RBF are shown in Table 2. The reward for tumbling was set to -10. A single episode was defined as a straight movement by 2.0 meters. The initial values were given randomly in the range of 0 to 1 for μ_{ij} and 1.0

for σ_j. The number of trial episodes was set to 2500 and the experiment was conducted using a simulator.

3.3 Experimental results

The trajetory of relocations by the robot during episode 1 and episode 2500 are shown in Fig. 7 as experiment results. The initial position of the robot was $(x, y) = (0, 0)$ according to the world coordinate system, facing the direction of the vector $(0, 1)$. The goal line is a straight line at $y = 2.0$. A comparison of the two trajectories reveal that the movements of the robot became near linear movement. Furthermore, there were little wasteful movements that were observed at the initial stage of the episode. The scene of convergence for the total reward value of the episode is shown in Fig. 8. It is evident that the total amount of reward has increased as the time progressed. The observed movements of the robot are shown in Fig. 9. The arm was rotated to the side of the failed crawler on the left side. The arm was thrust on the ground as if to lift the left crawler while the right crawler moved for the robot to advance straight ahead. The relocation trajectory depicted in Fig. 7 shows that the robot still needs the learning to improve the performance with regards to the relocation trajectory. It is evident that movements in linear directions is possible and the usefulness of the proposed method was confirmed.

4. CONCLUSION

The generation of a recovery motion involving a straight movement was set as the target with the assumption of a single crawler failure for a robot mounted with an arm. The dynamic simulator and the reinforcement learning were used to acquire the required motion. Rewards were set for relocation vectors, goals and robot stability to propose a rewards design method that would promote learning, which was evaluated through a computer experiment. The experiment results indicated that the linear relocation motion could be acquired when a crawler failed through a reinforcement learning by using the rewards design set for this research.

Future works include tuning of parameters relating to learning algorithm. Furthermore, evaluations under complex environment using multiple actuators would be necessary to verify the usefulness of the proposed method and evaluations using actual equipment would also be necessary.

REFERENCES

Hirose, S., Tsukagoshi, H., and Yoneda, K. (2001). Normalized energy stability margin and its contour of walking vehicles on rough terrain. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 1, 181–186. IEEE.

Ito, K., Kamegawa, T., and Matsuno, F. (2003). Extended qdsega for controlling real robots-acquisition of locomotion patterns for snake-like robot. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 1, 791–796. IEEE.

Kober, J. and Peters, J. (2012). Reinforcement learning in robotics: A survey. In *Reinforcement Learning*, 579–610. Springer.

Matsuno, F. and Tadokoro, S. (2004). Rescue robots and systems in japan. In *Robotics and Biomimetics, 2004. ROBIO 2004. IEEE International Conference on*, 12–20. IEEE.

Messuri, D. and Klein, C. (1985). Automatic body regulation for maintaining stability of a legged vehicle during rough-terrain locomotion. *IEEE Journal on Robotics and Automation*, 1(3), 132–141.

Murphy, R.R. (2004). Trial by fire [rescue robots]. *IEEE Robotics & Automation Magazine*, 11(3), 50–61.

Nakaoka, S. (2012). Choreonoid: Extensible virtual robot environment built on an integrated gui framework. In *System Integration (SII), 2012 IEEE/SICE International Symposium on*, 79–85. IEEE.

Platt, J. (1991). A resource-allocating network for function interpolation. *Neural computation*, 3(2), 213–225.

Strickland, E. (2014). Fukushima's next 40 years. *IEEE Spectrum*, 51(3), 46–53.

Sutton, R. and Gbarto, A. (eds.) (1998). *Reinforcement Learning*. MIT Press.