Reinforcement Learning-based Motion Generation for a Tracked Robot to Go Over a Sphere-shaped Non-fixed Obstacle

Hidenori Takamiya[1], Ryosuke Yajima[1], Jun Younes Louhi Kasahara[1], Ren Komatsu[1], Keiji Nagatani[1], Atsushi Yamashita[2] and Hajime Asama[1]

***Abstract*— Tracked robots have high traversability over rough terrain. However, even for such robots, it is still challenging to traverse terrain with non-fixed obstacles which may move when the robots go over them. Therefore, we propose a reinforcement learning-based method to generate the motion of the tracked robot to go over the obstacle. We set a task where the robot attempts to go over a sphere-shaped non-fixed obstacle and reach the goal. To succeed in the task, we designed a reward function so that the robot can reach the goal as straight as possible. As a training algorithm, Deep Q-Network was used and the robot was trained in a dynamics simulator. It was confirmed that the robot succeeded in the task using the trained network, which generated motion for going over a sphere-shaped non-fixed obstacle.**

I. INTRODUCTION

Every year, a plethora of disasters happen in Japan, such as earthquakes, volcanic eruptions, and typhoons. To mitigate the damage from those disasters, it is important to swiftly survey the affected area and to start disaster recovery as soon as possible. Because there are risks of secondary disasters, it is not desirable for humans to work onsite. Therefore, it is highly desirable that robots conduct the recovery tasks instead of humans. Robots can be distinguished by their locomotion system: wheeled robots, legged robots, jumping robots and flying robots have been considered for disaster recovery work. In this study, we focus on tracked robots due to their high traversability on rough terrain

Disaster areas are generally complex and are difficult to traverse even for tracked robots. One of the reasons is the existence of obstacles. There are many studies regarding tracked robots' ability to go over obstacles. For example, the changeable track mechanism was designed to enable the robot to ascend stairs [1][2]. In [3][4], by controlling sub-tracks adaptively with respect to the ground shape, it became possible for robots to run stably on rough terrain. Also, the terrain dynamics between ground and tracks, known as terramechanics, were considered to climb steps [5]. Aside modeling approaches, there are reinforcement learning-based approaches to run on rough terrain by selecting the optimal shape of tracks [6][7][8]. In those researches, the ability of tracked robots to go over obstacles was analyzed and improved. However, the target obstacles were fixed obstacles such as steps and stairs. In real disaster areas such as flooding or volcanic activity, there are mostly non-fixed obstacles such as loose rocks, which may move while interacting with the tracked robots.

[1]Hidenori Takamiya, Ryosuke Yajima, Jun Younes Louhi Kasahara, Ren Komatsu, Keiji Nagatani and Hajime Asama are with the Department of Precision Engineering, The University of Tokyo, 113-8656, Tokyo, Japan. takamiya@robot.t.u-tokyo.ac.jp

[2]Atsushi Yamashita is with the Department of Human and Engineered Environmental Studies, Graduate School of Frontier Sciences, The University of Tokyo, 277-8563, Chiba, Japan.

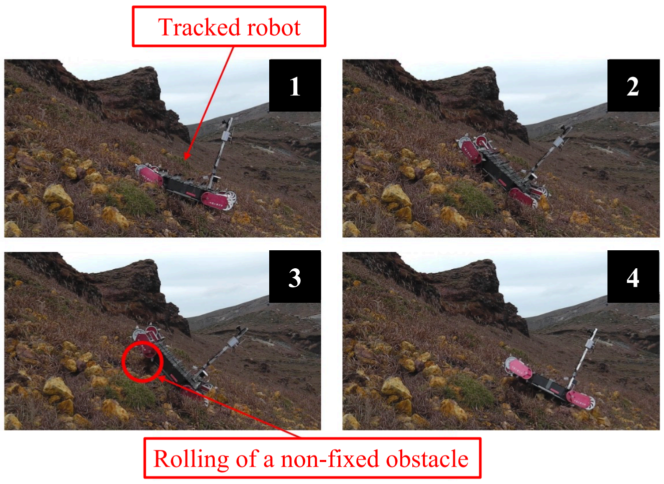
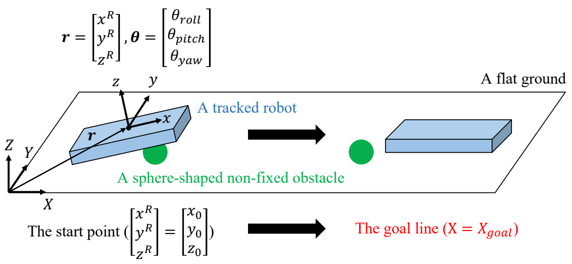


Fig. 1: The tracked robot fails to go over non-fixed obstacles. The robot climbs the obstacle from 1 to 2, and the obstacle rolls under the robot in 3, which causes the robot to slide down backward in 4.

Faced with such non-fixed obstacles, tracked robots may fail to go over them as illustrated in Fig. 1 because of the rolling of the obstacles. This may lead to not only damage to the robot but also failure to reach the desired destination.

Few studies have considered such non-fixed obstacles. In [9], the authors analyzed sliding-down and turning-over conditions when the tracked robot goes over a non-fixed cylinder-shaped obstacle on a slope. However, they did not explicitly achieve the motion by the tracked robot to allow going over such a non-fixed obstacle. Thus, the objective of this research is motion generation for a tracked robot to go over non-fixed obstacles.

In this paper, we proposed a reinforcement learning-based method to go over a sphere-shaped non-fixed obstacle by designing a reward function for a task as described in the next section. We confirmed the effectiveness of reinforcement learning-based approach and the tracked robot successfully generated the motion to go over the obstacle in a dynamics simulator.



$\boldsymbol{r} = \begin{bmatrix} x^R \\ y^R \\ z^R \end{bmatrix}, \boldsymbol{\theta} = \begin{bmatrix} \theta_{roll} \\ \theta_{pitch} \\ \theta_{yaw} \end{bmatrix}$
A flat ground
A tracked robot
A sphere-shaped non-fixed obstacle
The start point ($\begin{bmatrix} x^R \\ y^R \\ z^R \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}$)
The goal line (X = X_{goal})

Fig. 2: This study considers an environment where there is one sphere-shaped non-fixed obstacle on a flat ground. The task is to reach the goal line from the start point, in which the tracked robot is already on the non-fixed obstacle.

II. PROBLEM SETTING

In this study, for the sake of simplicity, we set an environment where there is one sphere-shaped non-fixed obstacle on a flat ground, as in Fig. 2. The task is to reach the goal line $X = X_{goal}$ from the start point. In the start point, the tracked robot is already on the non-fixed obstacle. Here, this study assumes an environment where obstacles densely exist on the ground and the tracked robot cannot make path planning that avoids climbing obstacles.

The coordinate systems fixed to the environment and the tracked robot are defined as XYZ and xyz, respectively. $\boldsymbol{r} = (x^R, y^R, z^R)^T$ is the vector of the origin of the tracked robot coordinate system, $\boldsymbol{\theta} = (\theta_{roll}, \theta_{pitch}, \theta_{yaw})^T$ is the vector of the tracked robot's orientation. It is assumed that these $\boldsymbol{r}, \boldsymbol{\theta}$ and their differential values are known, and the obstacle entry point relative to the tracked robot, $y = p$ is known.

III. METHOD

A. Learning-based Approach

There are model-based approaches, but those require experts to design a model for each environment setting. This process is very laborious, time-consuming and more importantly, tracked robots cannot perform well when faced with a new situation. To deal with this problem, it is better that the tracked robot learns optimal motions to succeed in tasks through interaction with the environment. Therefore, in this paper, reinforcement learning [10] is used.

B. Q-learning Formulation

A state, action, and reward at time step t are defined as s_t, a_t and r_t, respectively. $\pi(a_t|s_t)$ is called policy, which represents the probability distribution of a_t choice when the tracked robot is in state s_t. During training phase, policy $\pi(a_t|s_t)$ is epsilon greedy policy to explore an expectable action in low probability ϵ as follows:

$$\pi(a_t|s_t) = \begin{cases} 1-\epsilon & (a_t = \arg\max_{a'} Q(s_t, a')) \\ \epsilon & (a_t \text{ is chosen at random}) \end{cases}. \quad (1)$$

Q-learning is defined as maximizing the Q function $Q_\pi(s,a) = \mathbb{E}_\pi[G_t|s_t = s, a_t = a]$, where $G_t = \sum_{k=0}^{T-t} \gamma^k r_{t+k}$, $\gamma \in [0,1]$ and T is the final time step of each episode. G_t is the sum of discounted expected returns. $Q_\pi(s_t, a_t)$ is updated as below:

$$\begin{aligned} &Q(s_t, a_t) \\ &\leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)). \end{aligned} \quad (2)$$

Based on Eq. (2), the Q function is updated and converges towards the optimal Q function Q^*. The optimal policy π^* is deterministic policy and selects action $a_t = \arg\max_{a'} Q^*(s_t, a')$.

C. Deep Q-Network

Because Q function is represented as a table of states and actions, Q-learning cannot be applied directly to the continuous state space. We decided to use Deep Q-Network (DQN) [11], which applies a neural network to the Q function so that continuous state space can be inputted. Q function is approximated as Q_ϕ with parameter ϕ. Furthermore, DQN uses two techniques: experience replay and target network. Experience replay consists of adding an experience (s_t, a_t, r_t, s_{t+1}) to a replay buffer and sample mini-batches from the buffer. Target network consists in setting another Q network using parameter ϕ'. This parameter ϕ' is fixed and synchronized with the original Q network parameter ϕ every C episode.

D. Definition of State and Action

The state s_t is defined as $s_t = (\boldsymbol{r}, \dot{\boldsymbol{r}}', \boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, p)$. Here $\dot{\boldsymbol{r}}'$ is the velocity of the robot in the robot coordinate and is calculated using tranformation matrix from the world coordinate to the robot one. The action a_t is defined as 9 discrete actions which are combinations of driving right and left tracks with 3 discrete angular velocities, $0, v, -v$ [rad/s]. The definition of the discrete a_t number is shown in Table I.

Here, c_r and c_l represent angular velocity commands of right and left tracks, respectively. For example, in action 0, rotating right and left tracks both with the angular velocity v, generates forward motion of the tracked robot. Also, in action 3, rotating right and left tracks with the angular velocity v and $-v$ respectively, generates counter-clockwise neutral turn.

TABLE I: Definition of discrete action a_t.

a_t number	Commands to right and left tracks
0	$c_r = v$, $c_l = v$
1	$c_r = -v$, $c_l = -v$
2	$c_r = 0$, $c_l = 0$
3	$c_r = v$, $c_l = -v$
4	$c_r = 0$, $c_l = -v$
5	$c_r = v$, $c_l = 0$
6	$c_r = -v$, $c_l = v$
7	$c_r = 0$, $c_l = v$
8	$c_r = -v$, $c_l = 0$

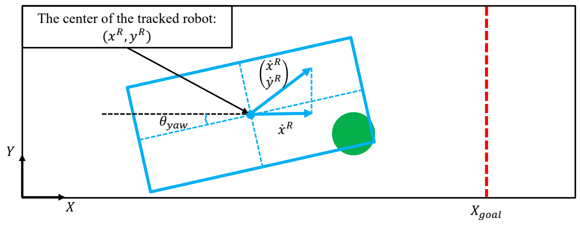

The center of the tracked robot:
(x^R, y^R)
$\begin{pmatrix} \dot{x}^R \\ \dot{y}^R \end{pmatrix}$
θ_{yaw}
$\dot{x}^R$
Y
X
X_{goal}

Fig. 3: Illustration of the environment from a top view. The light blue rectangle is the tracked robot and the green circle is the sphere-shaped non-fixed obstacle. The red dashed line represents the goal line.

E. Architecture of DQN

As an approximation function for the Q function, 4 fully connected layers with 128 units each are constructed. ReLU function is used as an activation function. In reinforcement learning, the system is usually considered in MDP framework. However this research problem is essentially a POMDP problem because the true position of the obstacle under the robot cannot be measured while going over the obstacle. To reduce the influence of POMDP, state is stacked for the past m experiences [11][12]. Thus the input of this network u is set $s_{t-i}(i = 0, 1, \ldots, m-1)$. In the experiment, m was set to 10. The output is $Q_{\phi}(s_t, a_t)$. The loss function is mean squared error with the output $Q_{\phi}(s_t, a_t)$ and target network $Q_{\phi'}(s_t, a_t)$.

F. Design of a Reward Function for the Task

As described in Section II, the task is that the tracked robot attempts to go over a sphere-shaped non-fixed obstacle and reach the goal. It is desired that the robot reaches the goal while keeping going as straight as possible. To reflect this idea into the reward function, at first, we set a positive reward if the robot is generating a larger velocity toward the goal. As shown in Fig. 3, the velocity toward the goal along X direction is represented as $\dot{x}^R$. If $\dot{x}^R$ is positive and large, the robot is considered to be driving smoothly toward the goal. If the maximum translational velocity of the robot is defined as $v_{\max}$, this reward r_{velocity} is expressed as follows:

$$r_{\text{velocity}} = \frac{\dot{x}^R}{v_{\max}}. \tag{3}$$

Second, we set a negative reward if the tracked robot deviates from a straight path to the goal. If $|\theta_{yaw}|$ is large, the robot deviates its desired straight direction as shown in Fig. 3. This reward r_{yaw} is defined as follows:

$$r_{\text{yaw}} = \frac{|\theta_{yaw}|}{\pi}. \tag{4}$$

Third, a reward related with the robot's pitch θ_{pitch} was introduced to avoid the robot going over the obstacle with a high pitch. The higher the robot's pitch, the more difficult it is to go over and more likely to slide down.

$$r_{\text{pitch}} = \frac{\theta_{pitch}}{\pi}. \tag{5}$$

From the above, we designed the reward function as follows:

$$r_t = w_1 \cdot r_{\text{velocity}} + w_2 \cdot r_{\text{yaw}} + w_3 \cdot r_{\text{pitch}} + w_4. \tag{6}$$

Here, w_i $(i = 1, 2, 3, 4)$ are hyperparameters that adjust the weight of each term. The fourth term in Eq. (6) is a penalty term for time steps to speed up the robot to reach the goal.

In addition, the reward r_T given at the end of each episode was set as follows:

$$r_T = \begin{cases} R_T & \text{if the goal was reached} \\ -R_T & \text{otherwise} \end{cases}. \tag{7}$$

IV. EXPERIMENT

In this experiment, the proposed method was implemented to the tracked robot in a dynamics simulator, and we validated its effectiveness for going over a sphere-shaped non-fixed obstacle. The expriment environment was constructed using ROS and Gazebo simulator.

The tracked robot's 3-dimensional model was made based on the tracked robot model proposed in [13]. The tracks' friction coefficient was set to 0.85. We set the value of action, the track rotational velocity v, to $2.0\,[\text{rad/s}]$.

The obstacle's radius was $0.035\,[\text{m}]$ and the mass is about $0.54\,[\text{kg}]$ (the density is $3.0\,[\text{g/cm}^3]$). The tracked robot has been confirmed to be able to go over the obstacle of this size through prior manual operation in the simulator. Also, it was confirmed that the tracked robot was not able to go over the obstacle when it was commanded to simply keep going straight forward.

In this experiment, the task described in Section II is considered, that is the tracked robot, already on the sphere-shaped non-fixed obstacle, starts an action, and attempts to go over the obstacle and reach the goal line. The initial position and orientation of the tracked robot were $(x^R, y^R, z^R)^T = (-0.235, 0, 0.14)^T$, $(\theta_{roll}, \theta_{pitch}, \theta_{yaw})^T = (0, -0.15, 0)^T$ and the goal line was set at $X_{goal} = 0.5$.

During training, we changed the obstacle's initial position along y, p at each episode. However we found from prior experiment that the training was challenging if p was changed along the whole width of the bottom surface of the robot. To speed up training, we divided the range of p into five segments, $D_i(i = 1, 2, 3, 4, 5)$, and trained for each segment. This is emperical considering that the motion strategy seemed to be different in each segment from the prior manual operation. Here, we trained the robot from D_1 to D_3 considering symmetry. D_1, D_2 and D_3 were set as $[3W/10, W/2], [W/10, 3W/10]$ and $[-W/10, W/10]$, respectively, where W is the width of the robot bottom surface.

The condition to end an episode is that the tracked robot reaches the goal line, or it significantly goes backward before reaching the goal or deviates laterally ($x^R < -0.6$ or $|y^R| >$

0.7), or the yaw gets too large ($|\theta_{yaw}| > \frac{\pi}{2}$). Episodes were ended when the time step was over 200.

The hyperparameters of this experiment are shown in Table II.

V. RESULT

The results of rewards during training in each segment are shown in Fig. 4. The light blue line shows the reward in each episode and the blue one shows the moving average reward every 100 episodes. The moving average reward in all segments was getting higher as the episode went on. However, the reward (plotted in a light blue line) is vibrating and it means that the robot was not be able to succeed in the task in some situations. The training result will be better by tuning hyperparameters and this is left as a future work.

Next, we analyze the actual motion generated by the trained Q network. Fig. 5 shows the result of the generated motion in segment D_3 and Fig. 6 shows the action a_t selected by the trained network at each time step[1]. Taken steps from the start point to the goal was 137, 1 step took about 0.1 [s], and the total taken time was about 14 [s].

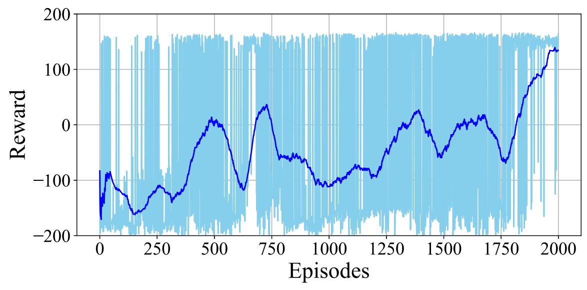
When the tracked robot was controlled to simply go forward, the robot was not able to go over the non-fixed obstacle and continuously slid down as shown in Fig. 5(a). This was because the obstacle under the robot started rolling before the center of the gravity of the robot went over the contact point between the robot and the obstacle.

On the other hand, from Fig. 5(b), the tracked robot operated two tracks and moved the obstacle to the edge of the robot's bottom surface, which led the robot to tilt. Then one track contacted the ground and went over the obstacle. Specifically, the tracked robot started an action at $t = 0$, and then the robot took the actions $a = 3$, which can be seen from Fig. 6. The action 3 rotated the right track v and the left one $-v$ from Table I and realized the turning motion. Due to this turning motion, the obstacle moved to the edge of the robot, which made the robot tilt at $t = 25$. The left track contacted the ground surface and could give the traction power from the ground to the front of the track. Then, the robot moved forward by $a = 7$ on the obstacle and went it over around $t = 50$. Then, the tracked robot was able to go over the obstacle at $t = 113$ and reached the goal line. In the prior experiment, the tracked robot was operated manually and it was not able to go over the non-fixed obstacle if it was kept going forward, using action $a = 0$. From this experiment, the proposed method itself did not select the forward action. Instead, actions which move the robot laterally and relatively moves the obstacle to the edge were selected as the optimal policy by the trained network.

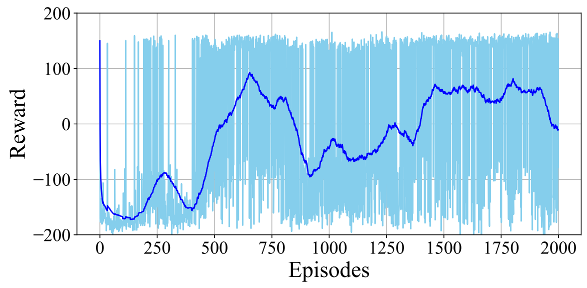
Examples of the results of the motion in segments D_1 and D_2 are made available online[2]. The optimal policy is almost same as the one in D_3, which moves the obstacle to the edge of the robot, then makes it tilt and contact the ground. However, the selected actions to turn the robot were different as shown in Fig. 7. This may be because the obstacle's motion such as rolling direction changes according to the track motion selected by the trained network. As a result, the trained network learned different actions in each initial

TABLE II: Hyperparameters used in this experiment.

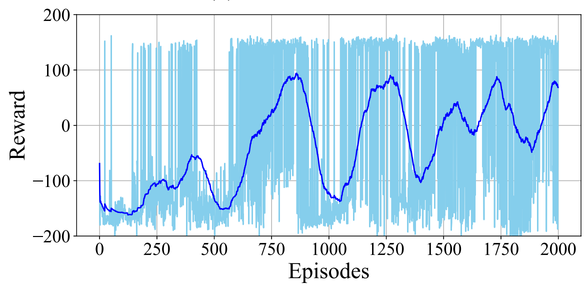
Hyperparameter	Value
w_1	1.0
w_2	-1.0
w_3	1.0
w_4	-0.1
ϵ	0.1
γ	0.98
C	20
R_T	100
learning rate	0.001



(a) Reward in D_1



(b) Reward in D_2



(c) Reward in D_3

Fig. 4: The result of rewards during training. The light blue line is the reward in each episode and the blue one is the moving average reward every 100 episodes. For readability, the axis of the reward is limited from -200 to 200. The reward grew higher in all segments as the episode went on.

[1]A video is available at: `https://youtu.be/oFCa_oh16u0`.

[2]`https://youtu.be/lXfPJogBmGk`. Note that the trained network in segment D_2 was the one saved at the 1900th episode.

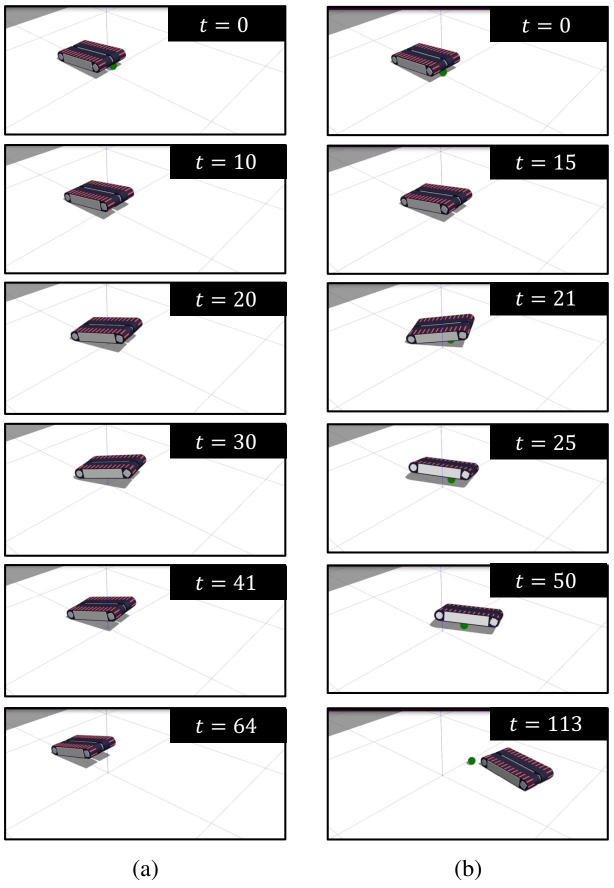


Fig. 5: Motion results. Fig. 5(a) shows the result where the tracked robot was simply kept going forward. The robot started to climb the obstacle. However the robot was not able to go over it. On the other hand, in Fig. 5(b), the tracked robot was able to go over using the actions selected by the trained network.

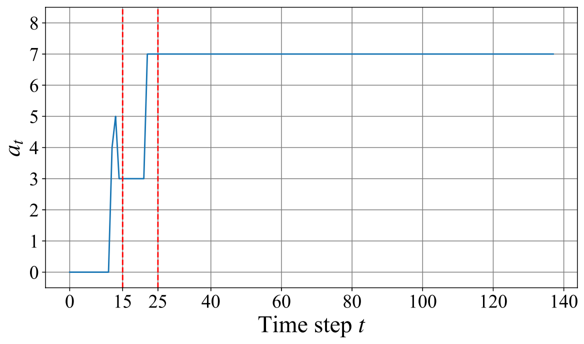


Fig. 6: The change of selected actions. The trained network selected action 3 to turn the robot and make it tilt. The vertical red lines correspond to the key frame of Fig. 5.

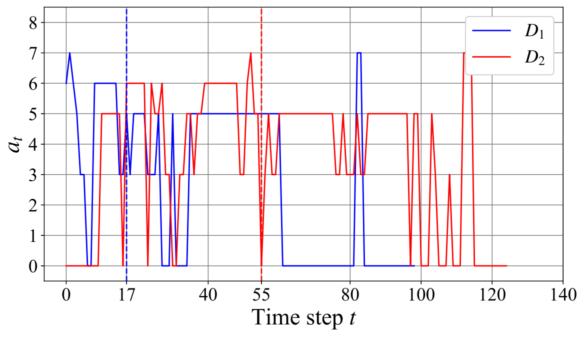


Fig. 7: The change of selected actions in segments D_1 and D_2. The vertical lines show the time at which the robot tilted.

position of the obstacle to realize the optimal policy.

VI. CONCLUSION

In this paper, we focused on tracked robots and generated motion for the robot to go over a sphere-shaped non-fixed obstacle. We used a reinforcement learning-based method called DQN, and designed a reward function appropriate for the task. The tracked robot was trained in a dynamics simulator, and we verified the effectiveness of reinforcement learning-based method for going over a sphere-shaped non-fixed obstacle in a simple environment.

Future work is constructing a method for more complex problem settings such as slope, rough terrain or going over other types of obstacles of different size, shape and friction. Actions will be continuous to adapt in various situations. The implementation of the trained network to the actual tracked robot is also necessary.

ACKNOWLEDGMENT

This work is supported by JST [Moonshot Research and Development], Grant Number [JPMJMS2032].

REFERENCES

[1] Lee C H, Kim S H, Kang S C, Kim M S, Kwak Y K: "Double-track Mobile Robot for Hazardous Environment Applications", Advanced Robotics, vol. 17, no. 5, pp. 447-459, 2003.

[2] Choi K H, Jeong H K, Hyun K H, Do Choi H, Kwak Y K: "Obstacle Negotiation for the Rescue Robot with Variable Single-tracked Mechanism", Proceedings of the 2007 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, pp. 1-6, 2007.

[3] Okada Y, Nagatani K, Yoshida K: "Semi-autonomous Operation of Tracked robots on Rough Terrain Using Autonomous Control of Active Flippers", Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2815-2820, 2009.

[4] Ohno K, Takeuchi E, Chun V, Tadokoro S, Yuzawa T, Yoshida T, Koyanagi E: "Rollover Avoidance Using a Stability Margin for a Tracked robot with Subtracks", Proceedings of the 2009 IEEE International Symposium on Safety Security and Rescue Robotics, pp. 128-133, 2011.

[5] Li L, Wang W, Wu D, Du Z: "Research on Obstacle Negotiation Capability of Tracked Robot based on Terramechanics", Proceedings of the 2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, pp. 1061-1066, 2014.

[6] Zimmermann K, Zuzanek P, Reinstein M, Hlavac V: "Adaptive Traversability of Unknown Complex Terrain with Obstacles for Mobile Robots", Proceedings of the 2014 IEEE International Conference on Robotics and Automation, pp. 5177-5182, 2014.

[7] Yehezkel L, Berman S, Zarrouk D: "Overcoming Obstacles with a Reconfigurable Robot Using Reinforcement Learning", IEEE Access, vol. 8, pp. 217541-217553, 2020.

[8] Totani M, Sato N, Morita Y: "Step Climbing Method for Crawler Type Rescue Robot Using Reinforcement Learning with Proximal Policy Optimization", Proceedings of the 2019 12th International Workshop on Robot Motion and Control, pp. 154-159, 2019.

[9] Yajima R, Nagatani K, Hirata Y: "Research on Traversability of Tracked robot on Slope with Unfixed Obstacles: Derivation of

Climbing-over, Tipping-over, and Sliding-down Conditions", Advanced Robotics, vol. 33, no. 20, pp. 1060-1071, 2019.

[10] Sutton R S, Barto A G: Reinforcement Learning: An Introduction, MIT press, 2018.

[11] Mnih V, Kavukcuoglu K, Silver D, Rusu A A, Veness J, Bellemare M G, Graves A, Riedmiller M, Fidjeland A K, Ostrovski G, Peterson S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg A, Hassabis D: "Human-level Control through Deep Reinforcement Learning", nature, vol. 518, no. 7540, pp. 529-533, 2015.

[12] Haarnoja T, Zhou A, Hartikainen K, Tucker G, Ha S, Tan J, Kumar V, Zhu H, Gupta A, Abbeel P, Levine S: "Soft Actor-Critic Algorithms and Applications", arXiv preprint arXiv:1812.05905, 2018.

[13] Okada Y, Kojima S, Ohno K, Tadokoro S: "Real-time Simulation of Non-Deformable Continuous Tracks with Explicit Consideration of Friction and Grouser Geometry", Proceedings of the 2020 IEEE International Conference on Robotics and Automation, pp. 948-954, 2020.