



Using sim-to-real transfer learning to close gaps between simulation and real environments through reinforcement learning

Yuto Ushida¹ · Hafiyanda Razan¹ · Shunta Ishizuya¹ · Takuto Sakuma¹ · Shohei Kato^{1,2}

Received: 15 April 2021 / Accepted: 18 October 2021
© International Society of Artificial Life and Robotics (ISAROB) 2021

Abstract

We attempt to develop an autonomous mobile robot that supports workers in the warehouse to reduce their burden. The proposed robot acquires a state-action policy to circumvent obstacles and reach a destination via reinforcement learning, using a *LiDAR* sensor. Regarding the real-world applications of reinforcement learning, the policies previously learned under a simulation environment are generally diverted to real robot, owing to unexpected uncertainties inherent to simulation environments, such as friction and sensor noise. To address this problem, in this study, we proposed a method to improve the action control of an Omni wheel robot via transfer learning in an environment. In addition, as an experiment, we searched the route for reaching a goal in a real environment using transfer learning's results and verified the effectiveness of the policy acquired.

Keywords *LiDAR* sensor · Reinforcement learning · Transfer learning

1 Introduction

Recently, the demand for workers in the distribution industry has been increasing owing to the increase in the handling amount of baggage due to the emergence of online shopping. However, there is a shortage of workers in the distribution industry because of the declining birthrate and aging population; hence, workers are required to work hard for long hours, which triggers several health problems, such as back pain and knee injuries. Therefore, there is an urgent need to improve labor efficiency and reduce the workload in the distribution industry. Autonomous logistic robots are expected to reduce the burden of workers and achieve sustainable growth for businesses. In existing distribution warehouses that do not support automation, distribution lines are set up in relatively small spaces for people to work. To

create a fully autonomous warehouse, a substantially large space is necessary. Therefore, a semi-automated logistics system that can adopt existing facilities will be required in the future in terms of cost. This study aims to develop a robot that can assist in transporting goods in a warehouse, thereby semi-automating the logistics system that carries baggage. Several robots that provide services via autonomous navigation have been developed (e.g., [1–3]). However, it is difficult to provide all the control rules for these autonomous mobile robots, and if the robots fall into unanticipated conditions, they cannot be correctly operated. In addition, these robots cannot handle friction, vibration, and noise, which are parameters that humans do not completely understand. To address this problem, this study utilizes a method called reinforcement learning [4], which can handle complex environments with unknown parameters, as well as achieve the goal, regardless of the problems, provided the agent is given a goal state and a reward. Reinforcement learning is also beneficial for mobile robot tasks, such as avoiding unknown obstacles and finding the shortest path to target position, although they might have several paths to the target [5]. In this study, we adopt reinforcement learning to enable the robot acquire behavioral control and reach a goal while avoiding obstacles. The robot used in this study is an Omni-wheel robot that can be rotated on the spot and moves in all directions such that it can be operated in a confined space made for workers.

This work was presented in part at the 26th International Symposium on Artificial Life and Robotics (Online, January 21–23, 2021).

✉ Yuto Ushida
ushida@katolab.nitech.ac.jp

¹ Graduate School of Engineering, Nagoya Institute of Technology, Nagoya, Japan

² Frontier Research Institute for Information Science, Nagoya Institute of Technology, Nagoya, Japan

In general, when reinforcement learning is used to control an real robot, a state-action policy is obtained in a simulation environment and the policy is used to control the robot. However, a challenge exists as the state and behavior of the simulator do not always match those of the real world owing to friction and sensor noise that exist in an real environment. Therefore, we proposed a method [6] that utilizes transfer learning to address this problem in our previous study. However, the study did not sufficiently consider the environmental differences between the simulation and real environments, and it also did not verify the effect exerted by the number of times of transfer learning. In this study, we verify the accuracy of the real robot control by learning with more randomness in the simulation environment and then compare the path finding performance between different number of times of transfer learning in an real environment.

The remainder of this paper is organized as follows. In Sect. 2, to realize robot control in the real environment, as a preliminary learning, we perform reinforcement learning in a simulation environment under conditions that consider environmental errors in correlation with an real environment. In the simulation environment, we introduce randomness to the learning by randomizing the initial position for more robustness in the real environment. In Sect. 3, the policy learned in the simulation environment is only used for the real robot, and the accuracy of the real robot control is measured. In Sect. 4, we adapt the policy to an real environment via transfer learning, control the real robot, and measure the accuracy of the real robot control. We also verify the control accuracy of the real robot depending on how many times the transfer learning was performed. In Sect. 5, we summarize this study and discuss future prospects.

2 Learning in the simulator

In this section, a robot model learns in a simulation environment. In this section, a robot model is learned in a simulation environment. The robot is modelled as an omni-wheel robot, which can move in all directions while rotating on the spot to operate in a limited space for workers. We adopt Q-learning [7] as a reinforcement learning algorithm, which has the advantage of learning convergence. Algorithm 1 presents Q-learning algorithm.

Algorithm 1 Q-Learning algorithm

Algorithm parameters: step size $\alpha \in (0,1]$, small $\epsilon > 0$
Initialize $Q(s,a)$, for all $s \in S^+, a \in A(s)$, arbitrarily except that $Q(\text{terminal}, *) = 0$
for episode = 1, M **do**
 Initialize S
 for step = 1, N **do**
 Choose A from S using policy derived from Q (e.g., ϵ -greedy)
 Take action A , observe R, S'
 $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S,a) - Q(S,A)]$
 $S \leftarrow S'$
 end for
end for

Learning in the simulation environment was conducted in two phases. First, the robot learns from a fixed start point in 1000 episodes. Second, the robot learns from a random start point in 2000 episodes. We expect this phasing to make the robot more robust in real environments. In addition, the action-value function $Q(s, a)$ in Q-learning has been updated in the following equation:

$$Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1}, a_{t+1})). \quad (1)$$

where α , γ , s_t , a_t , and r_t denote the learning rate, discount rate, state, action, and reward, respectively. In this experiment, the training parameters α and γ were set to 0.2 and 0.8, respectively. The state of the robot model in the experiment was defined using the value of a *LiDAR* sensor. The robot model adopted in the experiments is an Omni-wheel robot with a *LiDAR* sensor. Figure 1 illustrates the system configuration of the learning experiment conducted in the simulation environment. The experiment adopts a robot operating system (ROS) as the control system for the robot model. Gazebo, a 3D simulator, simulates a mobile robot with a 270 degree *LiDAR* sensor. OpenAI supports the learning calculation and generates the policy to be applied in the actual wheeled robot.

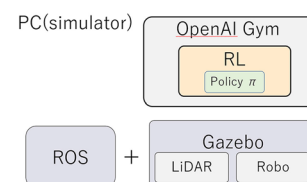
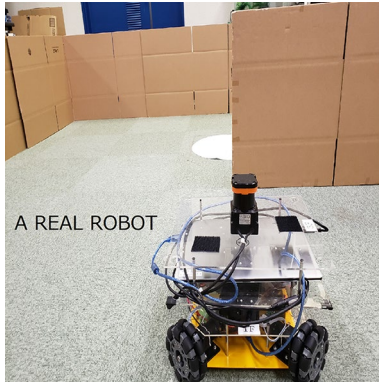
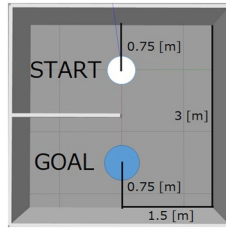


Fig. 1 System configuration of the simulator

Fig. 2 Simulator map**Fig. 3** A real robot

2.1 Simulation conditions

2.1.1 Map

Figure 2 shows a map used in the learning. The white and the blue circles in a map show the start and the goal points, respectively, of the simulation. The map is 3 m squared, with the goal points of 0.25 m radius.

2.1.2 Robot

In this study, we develop policies in a simulator, which will be later transferred to an real robot (Fig. 3). Therefore, we need to create a 3D model robot that mimics the characteristics of a real robot. The simulator robot are illustrated in Fig. 4. The specifications of the real robot are presented in Table 1, while Fig. 3 shows the real robot.

The robot works with *Arduino 328*, a microcontroller with an I/O extension board for sensors and motors. The robot also uses a *LiDAR* sensor as a distance sensor, and its specifications are presented in Table 2.

The *LiDAR* sensor is mounted on the upper center of the robot, with a scanning angle of 270 degrees and a maximum range of 30 m. Based on these characteristics of the robot and the *LiDAR* sensor, a 3D model of the robot was created in the simulator.

Table 1 Omnidirectional mobile robot specifications

Robot name	NEXUS robot (4WD)
Maximum length	402 mm
Height	100 mm
Maximum motor velocity	0.6 m/s
Power consumption	17 W
Weight	4 kg
Maximum load	5 kg

Table 2 *LiDAR* sensor specifications

Sensor name	<i>LiDAR</i> Hokuyo UTM-30LX
Scanning range	0.1 ~ 30 m
Scanning angle	270°
Scanning time	25 ms/scan
Power source	12 VDC \pm 10%
Current consumption	700 mA

2.1.3 States

Figure 5 presents the five ranges of *LiDAR* as states: 0–1[deg], 66.5–68.5[deg], 134–136[deg], 201.5–203.5[deg], and 269–270[deg]. The states used for learning take the median of these ranges and are discretized by 0.25 m in each of the five directions from 0 to 4.25 m. The states for the Q-learning are the combination of the five discretized data presented below the data. When the sensor data are 0.25 m, the discretized value is considered 0.5 m, and the value on the dividing line is rounded up.

$$\text{Discretizeddata} = \{0.0, 0.25, 0.5, \dots, 3.75, 4.0, 4.25\} \quad (2)$$

2.1.4 Actions

The robot has four actions: front, back, right, and left actions, and the speed is set to 0.25 m/s. In one action, the robot moves for 0.5 ± 0.01 s of the simulator time.

2.1.5 Rewards

The robot gets a reward for every step presented in Table 3. This experiment has four reward conditions in the learning. For the first one, if the robot model hits the wall, it

Fig. 4 3D model of the omnidirectional mobile robot

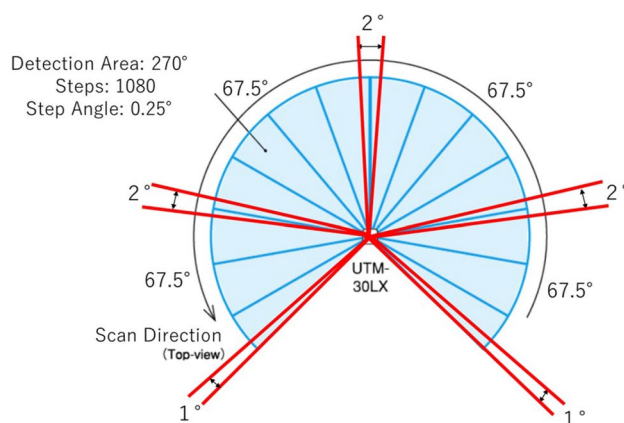
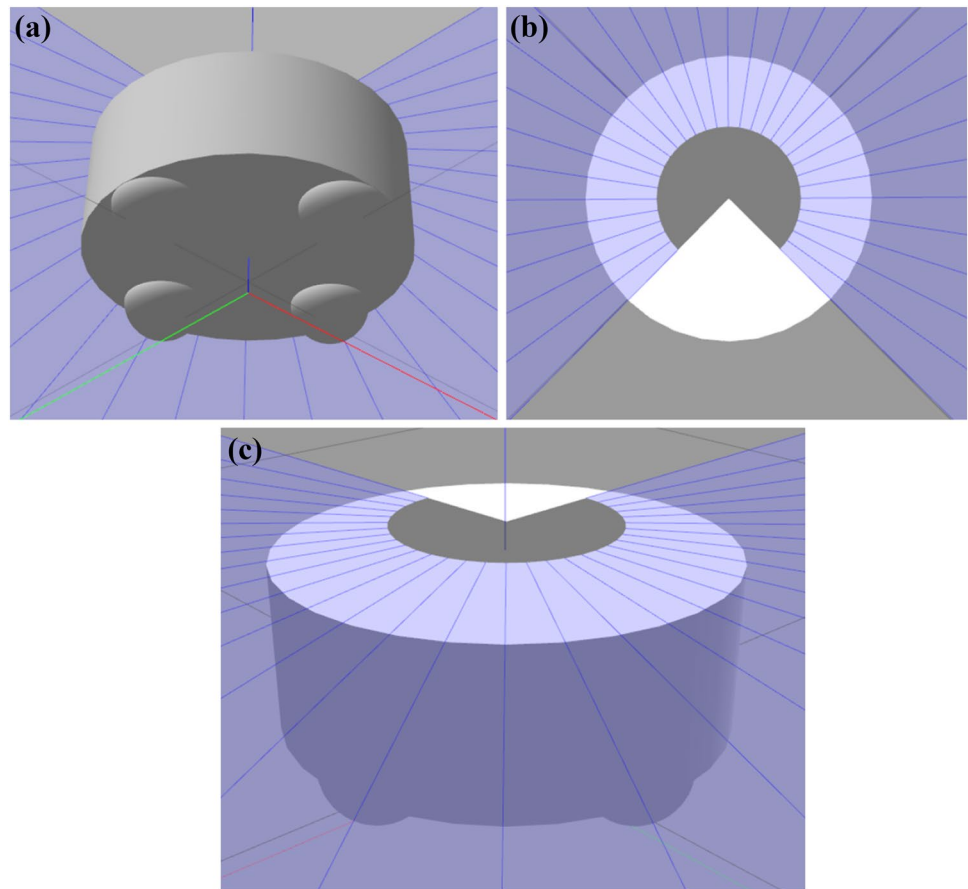


Fig. 5 LiDAR data used for the robot state

receives -10^6 reward points. Second, if the robot model is in the same distance as before or moves further from the goal point, it receives -10 reward points. Third, if the robot model moves closer to the goal point, it receives -1 reward points. Fourth, if the robot model reaches the goal, it receives 0 reward points.

Table 3 Rewards

Condition	Reward
Contact the wall	- 1,000,000
Same distance to the goal	- 10
Away from the goal	- 10
Close to the goal	- 1
Arrive at the goal	0

Table 4 Learning

Initial position	Number of episodes	success rate
Fixed + random	1000 + 2000	96.6 [%]

2.2 Result of the simulator

Table 4 presents the results obtained from the learning experiment in the simulator. Figure 6 illustrates one of the fewest actions in the entire experiments in the map. Based on these results, we can infer that by adopting Q-learning, which is one of the basic types reinforcement learning, a

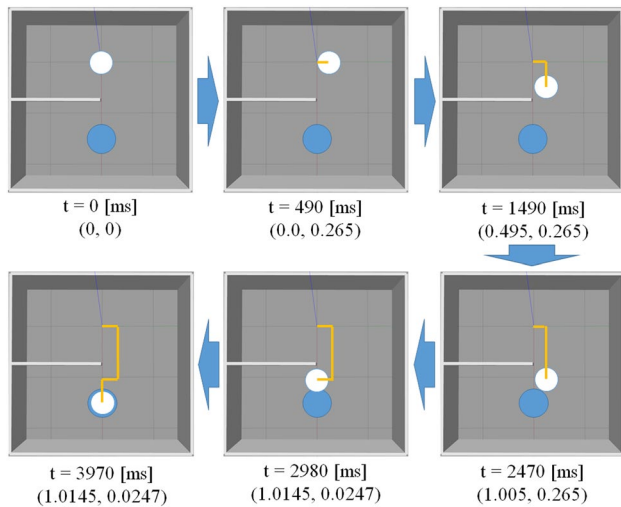


Fig. 6 One of the fewest actions

Fig. 7 A sensor (UTM-30LX)



mobile robot can determine a path to the goal while avoiding static obstacles.

3 Real-world application

The learning results obtained from the simulator are applied to the actual omni-wheel robot presented in Fig. 4. The *LiDAR* measurements are adopted as the real-world conditions, shown in Fig. 5, as well as the simulator learning. A *LiDAR* sensor, UTM-30LX [8], illustrated in Fig. 7, is mounted on the robot. The real robot selects an action whose value is maximum in a given state. It selects action randomly when the robot reaches an inexperienced state. Figure 8 presents a system configuration of the experiment conducted in the real environment.

3.1 Map

Figure 9 presents a real-world map corresponding to the map shown in Fig. 2. The map is 3 m square environment. The robot started at 1.5 m from the leftmost wall and 0.75 m from the wall behind.

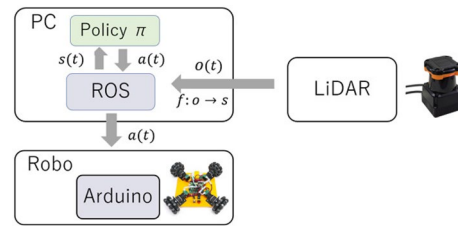


Fig. 8 System configuration of the real environment

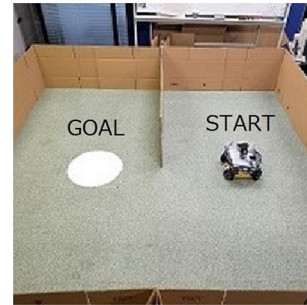


Fig. 9 Real map

3.2 Naive transfer results

First, we simply applied the policy from the simulator to the real robot and tested the success rate with 30 trials for the setting. Accordingly, the robot achieved a success rate of 40.0 [%] of dropping 56.6 [%] from the result of the simulator experiment, and this result was completely unacceptable.

4 Transfer learning

Transfer learning is a framework in which knowledge learned by the source agent is reused by the destination agent. Recently, it has been applied to machine learning and reinforcement learning with good results. In transfer learning within the framework of reinforcement learning, a reinforcement learning agent learns and acquires a policy at the source. Then, in the same or similar environment, the same or similar agent can reuse the policy acquired at the source to speed up the acquisition of the solution at the destination [9]. Taylor et al. defined transfer learning in reinforcement learning in [10, 11].

$$Q_c(s, a) = Q_d(s, a) + Q_s(\chi_s(s), \chi_a(a)), \quad (3)$$

where $Q_s(\chi_s(s), \chi_a(a))$ is the policy transferred from the source, $Q_d(s, a)$ is the policy used by the agent at the destination, and the action values learned in the new environment are also updated here. Further, $Q_c(s, a)$ denotes the policy that combines the transferred policy and the policy acquired

at the destination, and the action is selected at the destination using the policy $Q_c(s, a)$. The function $\chi_x(x)$ is an Inter-task mapping that describes the correspondence of x between sources and destination. In this study, for a policy adaptation using transfer learning, a real robot can reuse the policy acquired by robot model in the simulator environment, by defining $\chi_x(x) = x$, that is, considering same dimensions between simulation and real world for states and actions.

This transfer learning is used to address the problem that the policy learned in the simulator cannot be used effectively owing to unexpected uncertainties under the simulation environment, and the effectiveness of transfer learning in the real operation of reinforcement learning was verified. In this study, we performed transfer learning up to 50 episodes using Eq. 3 to improve the policy learned under simulation. After learning, we conducted experiments in real environments using the refined policy at different episodes to measure the success rate of reaching the goal. Subsequently, we verified the effectiveness of the transfer learning by comparing the success rates using policy refined at the 0th, 10th, 20th, 30th, 40th, and 50th episodes. We also examined the changes that occurred as the transfer learning progressed by comparing the robot's trajectory with the unlearned position. The learning rate α , in Eq 3 was set to 0.1.

4.1 Application results

Figure 10 presents the success rate for each of the 0th, 10th, 20th, 30th, 40th, and 50th episodes of transfer learning. The results indicate that the success rate is improved by up to 50 points through transfer learning. Figure 11 shows the number of learned positions each of the 0th, 10th, 20th, 30th, 40th, and 50th episodes of transfer learning. Figures 12(a)-(f) present the trajectories of the robot that moved according to the policy, each of the 0th, 10th, 20th, 30th, 40th, and 50th episodes of transfer learning.

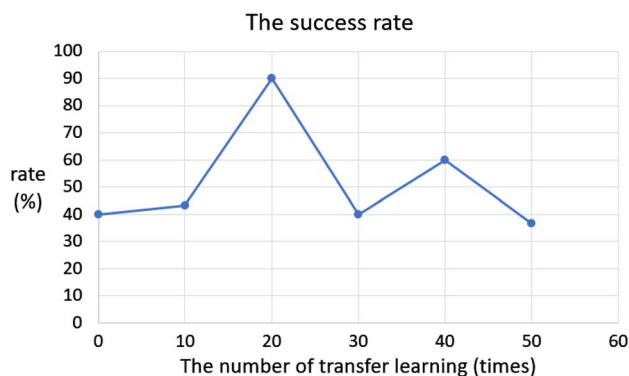


Fig. 10 Results of transfer learning

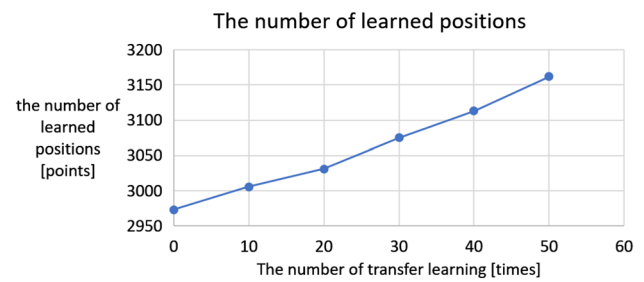


Fig. 11 Number of learned positions

4.2 Discussion

Even if randomness is introduced to the learning in the simulation environment, the real robot may reach an inexperienced state, provided that it uses the policy learned in the simulation environment. In this study, we used the policy acquired by adding randomness to learning in the simulation

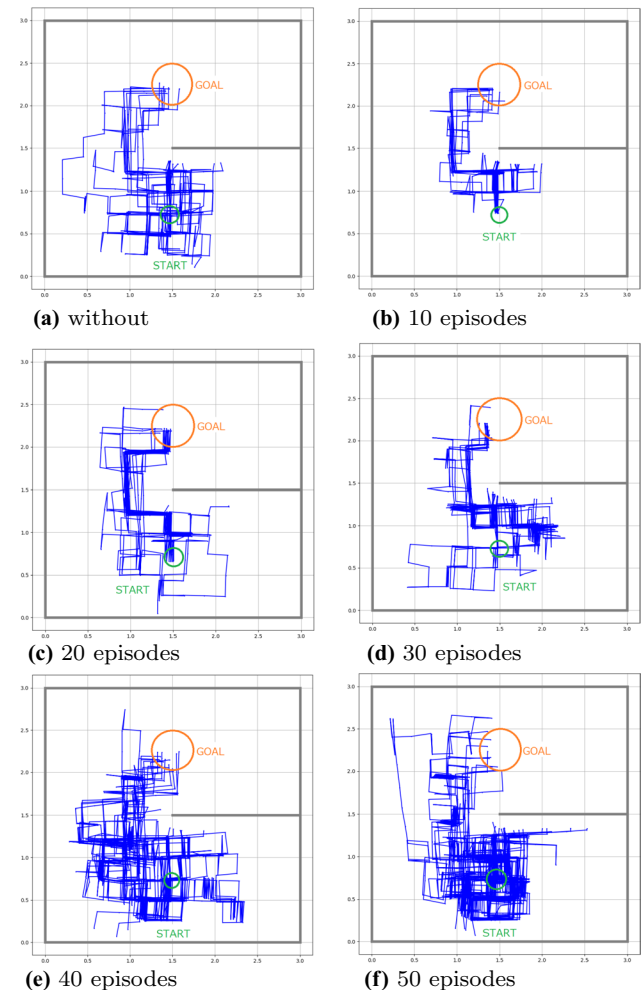


Fig. 12 Trajectories of the real robot using policy refined by transfer learning

environment, such as randomizing the initial position, and conducted transfer learning in the real environment. Accordingly, the behavioral trajectories shown in Figs. 12(a)–(c) and Fig. 10 indicate that learning progressed smoothly up to 20 episodes of transfer learning, with the highest success rate at the 20th episode. We believe that the reason for this improved success rate is that transfer learning reduced the number of inexperienced states that could not be considered in the simulation environment. In contrast, the observation of the entire behavioral trajectory shown in Figs. 12(a)–(f) and Fig. 10 suggested that learning stuck after the 20th episode of transfer learning and may start to find a shorter path. Figure 11 indicates that the number of learned positions increases linearly. However, the transition in success rate shown in Fig. 10 is not monotonic, as increasing the number of learned positions.

5 Conclusion

To develop a working robot in warehouses by adopting reinforcement learning, it is necessary to reduce the environmental error between the simulation environment and the real environment. This paper proposed a method of sim-to-real transfer learning to close gaps between simulation and real environments. In this study, we verified the accuracy of the real robot control by learning with more randomness in the simulation environment and then compared the path finding performance between different number of episodes of transfer learning in the real environment. Accordingly, the accuracy of behavioral control of the real robot was greatly improved by transfer learning compared to the case without transfer learning, and the difference in the success rate of reaching the goal was verified depending on the number of episodes of transfer learning. However, because the experiments in this study have only verified the effectiveness of transfer learning in one environment, it is necessary to verify the effectiveness of transfer learning in more complex environments in the future.

In this study, the simulator environment did not model any type of noise in the real environment, such as friction and sensor noise. To address this problem in the future, we will focus on introducing this type of noise to the state and control values in the learning in the simulation environment to consider the environmental error between the real and simulation environments, as well as focus on sophisticated inter-task mapping $\chi_x(x)$ (see Eq. 3).

References

1. Gen A (2006) Development of outdoor cleaning robot SUBA-RUROBOHITER RS1. *J Robot Soc Japan* 24(2):153–155
2. Yuzi H (2006) An approach to development of a human-symbiotic robot. *J Robot Soc Japan* 24(3):296–299
3. Yoichi S et al (2006) Security services system using security robot Guardrobo. *J Robot Soc Japan* 24(3):308–311
4. Sutton RS, Barto AG (1998) *Reinforcement Learning*. The MIT Press, Cambridge
5. Sunayama R et al. (2007) Acquisition of collision avoidance behavior using reinforcement learning for autonomous mobile robots ROBOMECH lecture, pp.2A1-E19-1-4
6. Yuto U et al. (2020) Policy transfer from simulation to real world for autonomous control of an omni wheel robot, proceedings of 2020 IEEE 9th Global Conference on Consumer Electronics, pp.170-171, October 13-16
7. Watkins CJCH, Dayan P (1992) Technical note: Q-learning. *Mach Learn* 8:56–68
8. Laser Range Scanner Data Output Type/UTM-30LX Product Details IHOKUYO AUTOMATIC CO.,LTD. Avail-able: <https://www.hokuyo-aut.co.jp/search/single.php?serial=21>
9. Lazaric A (2012) Transfer in Reinforcement Learning: a Framework and a Survey, *Reinforcement Learning-State of the art*. Springer, Berlin, pp 143–173
10. Taylor ME, Stone P (2009) Transfer learning for reinforcement learning domains: a survey. *J Mach Learn Res* 10:1633–1685
11. Taylor ME (2009) Transfer in reinforcement learning domains. *Studies in computational intelligence*, vol. 216. Springer, New York

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.