ADVANCED ROBOTICS
2022, VOL. 36, NO. 8, 404–421
https://doi.org/10.1080/01691864.2022.2046504

FULL PAPER

Residual reinforcement learning for logistics cart transportation

Ryosuke Matsuo[a], Shinya Yasuda[b], Taichi Kumagai[b], Natsuhiko Sato[b], Hiroshi Yoshida[b] and Takehisa Yairi[a]

[a]Department of Aeronautics and Astronautics, The University of Tokyo, Tokyo, Japan; [b]System Platform Research Laboratories, NEC Corporation, Kanagawa, Japan

ABSTRACT

Autonomous logistics cart transportation is a challenging problem because of the complicated dynamics of the logistics cart. In this paper, we tackle the problem by using two robots system with reinforcement learning. We formulate the problem as the problem of making a logistics cart track an arc trajectory. Our reinforcement learning (RL) controller consists of a feedback controller and residual reinforcement learning. The feedback controller regards a logistics cart as a virtual leader and robots as followers, and the robots' position and velocity are controlled to maintain the formation between the logistics cart and the robots. Residual reinforcement learning is used to modify the other model's output. Simulation results showed that the residual reinforcement learning controller trained in a physical simulation environment performed better than other methods, especially under the condition with a large trajectory curvature. Moreover, the residual reinforcement learning controller can be transferred to a real-world robot without additional learning in a real-world environment.

ARTICLE HISTORY
Received 13 June 2021
Revised 8 November 2021
Accepted 4 February 2022

KEYWORDS
Reinforcement learning; logistics

1. Introduction

Object transportation is increasingly being automated through the use of automated guided vehicles (AGVs) in large warehouses. However, this is less common in smaller warehouses, where objects are typically conveyed by human workers with logistics carts because existing automation systems by AGVs are not supported to transport existing logistics carts. For example, a space below a logistics cart is too small to move under and lift it. To address this, an automated object transportation system for these warehouses [1] was proposed. In this system, the robot's position is estimated by utilizing images from a camera on the ceiling, and two robots grasp a logistics cart and transport it as shown in Figure 1. The strategy of having two robots hold a logistics cart makes it possible to automate the transportation without additional equipment. However, control for transporting a logistics cart remains a difficult problem because robots need to keep holding the cart. There is currently no method for making a logistics cart track a trajectory.

Reinforcement learning (RL) has enjoyed success in many gaming tasks and shows promise for constructing a controller that can adapt to controlling an agent in complex dynamics. Several deep reinforcement learning (DRL) methods for continuous control [2–7] have yielded impressive results in various robotics applications. However, as RL for robotics requires experiences to be gathered in the real world, the cost can become expensive. One solution is to use a physical simulator such as MuJoCo [8], Pybullet [9], V-REP [10], Gazebo [11], or Unity [12]. These improve the learning efficiency and reduce costs for managing robot safety. Furthermore, combining these simulators with a distributed reinforcement learning method [4,13–15] accelerates the learning speed. Distributed reinforcement learning is a method where multiple environments are simulated in parallel, so a high number of diverse experiences can be obtained efficiently.

In the context of control by DRL, unique methods such as residual reinforcement learning [16–18] and control structured policy learning [19], which include a feedback control structure, have been proposed. These methods improve the learning efficiency by utilizing prior knowledge.

In this work, we propose a method for constructing a residual reinforcement learning controller that modifies the base controller by reinforcement learning for the logistics cart transportation along a given trajectory. We do not consider trajectory planning. Our main contributions are as follows.

- We achieve logistics cart transportation by a reinforcement learning controller.

CONTACT Ryosuke Matsuo ryosuke129@g.ecc.u-tokyo.ac.jp

© 2022 Informa UK Limited, trading as Taylor & Francis Group and The Robotics Society of Japan

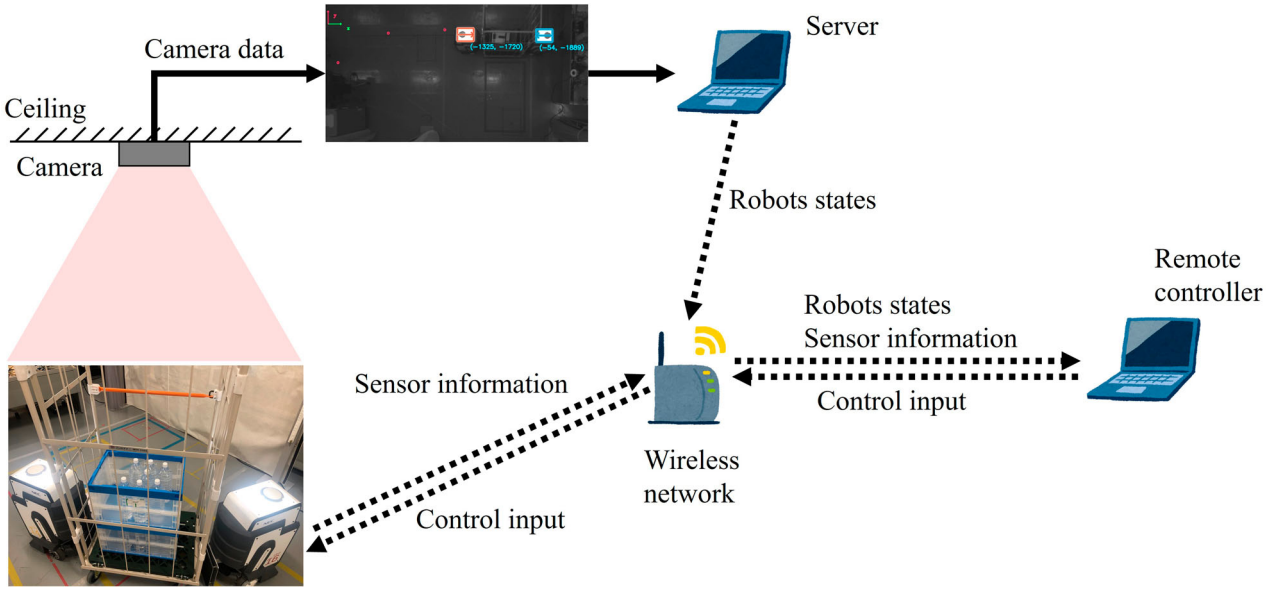

Figure 1. System overview.

- We utilize the physical simulator and knowledge of the feedback controller to make the controller learn the policy efficiently.
- Our proposed controller learned by physical simulation can be transferred to real-world robots without additional parameter tuning.

2. Related works

2.1. Deep reinforcement learning for cooperative multi-agent control

Deep reinforcement learning methods that deal with high-dimensional observation for multi-agent control have been extensively researched (see [20] for a survey). Much research has focused on discrete action space, with less attention paid to continuous action space.

Lowe et al. [21] implemented an actor-critic model in a cooperative multi-agent system where each agent has an independent Q-function whose inputs include its own observation, other agents' observations, and the policy. In the training phase, each agent learns the Q-function with the other agents' information and the policy learned using this Q-function. Then, in the execution phase, each agent utilizes the policy with only local information.

Gupta et al. [22] compared three strategies and learning algorithms in various simulations that request discrete action or continuous action at first. The three strategies are (i) Centralized, where one model includes joint observation and joint action, (ii) Concurrent, where each agent has an independent observation and model, and (iii) Parameter Sharing, where each agent has an independent observation and sharing model parameter. The authors found that Parameter Sharing was the most scalable to the number of agents and proposed Parameter Sharing TRPO (PS-TRPO). TRPO is a method of deep reinforcement learning for continuous control [3]. In our system, two robots are controlled as a single agent, however, it is related to centralized strategy in the context of multi-agent reinforcement learning.

2.2. Cooperative object transportation

Cooperative object transportation has been researched since the 90s. The strategies for solving this problem are mainly divided into three groups: the pushing strategy, the grasping strategy (where the object is fixed between robots by some equipment or is placed directly on the robots), and the caging strategy [23]. The pushing-only strategy appears simple strategy, however, robots can only push an object, therefore, the system requires delicate control. The grasping strategy has physical connection between robots and an object. Thus, it is easy to realize stable transportation and robots can exert pulling power on an object. However, physical connection is required. The caging strategy can realize stable transportation by caging an object by robots, however, a controller is required to keep caging an object and caging an object by a few robots is difficult. We adopt the caging strategy because of two reasons: (i) a logistics cart has casters, therefore, it is easy to keep moving due to inertia, and (ii) our system is introduced to existing warehouses and transports existing logistics carts, thus, an additional attachment for physical connection is undesirable.

Research by Ohsaki et al. [24] utilizes the pushing strategy with the object on casters. Two cooperating robots arrange small dollies under an object and then one robot pushes the object for transportation. This research focused on arranging the dollies and stabilizing the pushing robot, and did not discuss the details of the object transportation. Our research differs in that we focus specifically on logistics cart transportation. Another difference is that the weight of the object in [24] was 35 kg, whereas ours is considerably heavier.

As for the grasping strategy, some studies have examined leader-follower systems with two mobile robots [25–27]. In a leader-follower system, the leader tracks a given trajectory and the follower follows the leader for transporting the object. Extended methods where the object is regarded as a virtual leader and the robots are followers have been proposed [28].

Brown et al. [29] proposed a simple controller for cooperative object transportation by using the caging strategy with two robots. One of the robots guides the object movement, and the other pushes the object. Their research assumes that the contact point between the robots and the object slides for making the guidance robot track an arc trajectory, while in contrast, our system assumes that the contact point between the logistics cart and the robots does not slide. Moreover, the object property in [29] is different from that in our research.

With regard to the caging strategy, leader-follower systems that utilize three or more robots have been proposed. Wang et al. proposed a system featuring one leader and some followers [30]. Wan et al. [31] proposed a system where the object is regarded as the leader and robots are followers. Wan et al. developed a leader-follower system featuring a multi-fingered mechanism [32].

Methods for caging a concave object by means of a two-fingered mechanism have also been proposed [33–35]. See the work by Makita et al. for a detailed survey of the methods utilizing the caging strategy [36].

A reinforcement learning controller for cooperative object transportation by the grasping strategy has been proposed [37]. In this method, the controller is constructed by a deep Q network (DQN) that outputs a discrete action. Two robots and an object are arranged in a fixed environment and their purpose is to arrive at an exit. This research differs from our own in many respects, including the property of the object, the method of transportation, the action space, and the concept underlying the processing observation.

While much research has examined cooperative object transportation, there has been almost no research on the transportation of a logistics cart by two robots using the caging strategy, to the best of our knowledge.

3. Methodology

3.1. Hardware architecture

The hardware architecture is shown in Figure 2. The plate that contacts a logistics cart has a spring mechanism to enable flexible holding. This plate is covered by a high-friction material to avoid slippage between the robot and the cart. The holding mechanism has a spring-damper characteristic that rotates to track a circular trajectory. It stays at the center when it is in the free state, as shown in Figure 2 (a).

This system utilizes two robots: a supporting robot that supports a change in the direction of the logistics cart, and a pushing robot that pushes the cart. These roles are fixed when the logistics cart transportation is started and do not change during transportation.

3.2. Problem statements

3.2.1. Observation

The position and orientation of the robots are estimated by the image from the ceiling camera, which utilizes rectangular recognition. The robot position corresponds to the center of rotation and the installation point of the holding mechanism. The position and orientation of the logistics cart, in contrast, cannot be estimated by the image because of the complexity of the logistics cart and the loading object.

If the controller utilizes the robots position and orientation in global coordinates, it does not work when the scale of the environment changes. Therefore, we utilize local coordinates calculated by the estimated position and orientation of the logistics cart. Figure 3 shows the ideal arrangement of the robots and logistics cart in local coordinates. The orange line is the ideal trajectory of the logistics cart and is given in advance, i.e. we do not consider trajectory planning. The true position and orientation of the logistics cart is calculated by the pushing robot position, orientation, and rotation angle of the holding mechanism. The y-axis of local coordinates corresponds to the line from the logistics cart position to the center of the circular trajectory and the x-axis is the tangent of the circular trajectory. The position and orientation of the robots in these local coordinates is the observation of the reinforcement learning controller. However, in real situations, robots often cannot maintain the ideal holding, so the position and orientation of the logistics cart become estimated values and local coordinates are calculated by the estimated position and orientation. The reason we only use the pushing robot for calculating the logistics cart position and orientation is that the pushing robot is required to contact the logistics cart for transporting it,

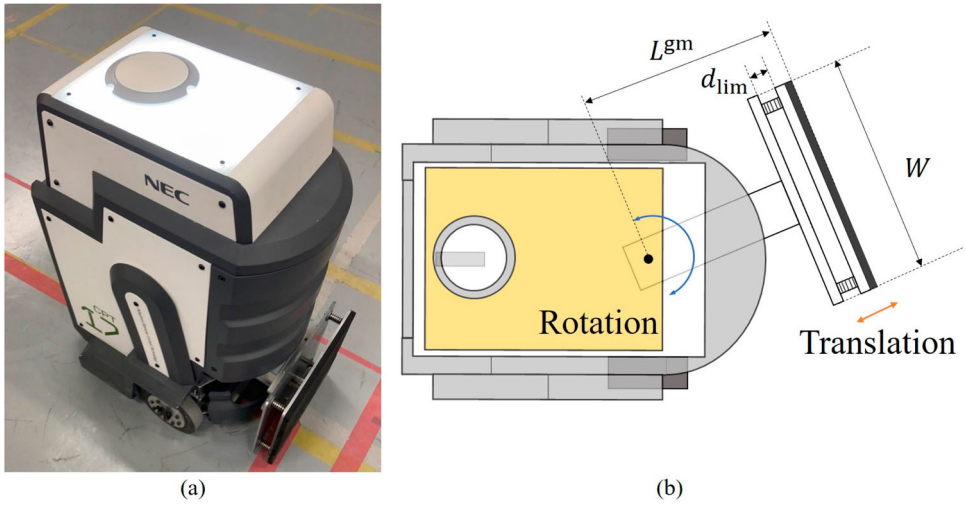

Figure 2. Hardware architecture: (a) prototype and (b) holding mechanism.

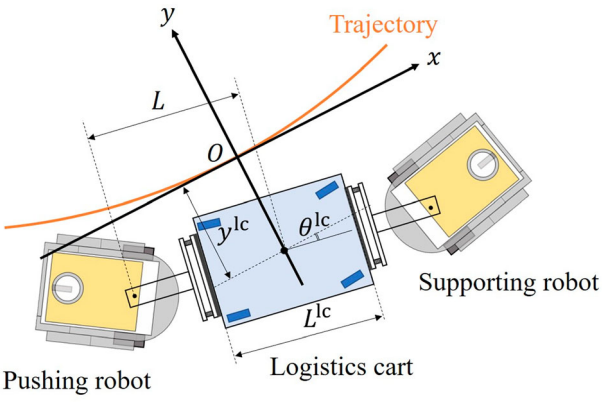

Figure 3. Local coordinates for calculating observation.

which means the estimated logistics of the cart position and orientation are calculated more accurately.

Additionally, the variation of each robot's touch plate and the rotation angle of the holding mechanism are included in the observation, and time differentials of these observations are added. Finally, an action before one step, the trajectory curvature, and the size of the logistics cart are appended to the observation.

3.2.2. Action

The continuous action space is defined as each robot's linear velocity and rotational velocity. Therefore, the action space has four dimensions. The range of the linear velocity of each robot is $[-0.5, 0.5]$ m/s and the range of the rotational velocity of each robot is $[-\pi/6, \pi/6]$ rad/s. The observation variables and action variables are listed in Table 1.

Table 1. Observation variables and action variables. FB obs. means the observations for the feedback controller.

Observation variables	Symbols	FB obs.
Supporting robot position	x^{sr}, y^{sr}	✓
Supporting robot orientation	$\cos\theta^{sr}, \sin\theta^{sr}$	✓
Pushing distance of grasping mechanism of supporting robot	d^{sr}	
Rotation angle of rotation mechanism of supporting robot	ϕ^{sr}	
Pushing robot position	x^{pr}, y^{pr}	✓
Pushing robot orientation	$\cos\theta^{pr}, \sin\theta^{pr}$	✓
Pushing distance of grasping mechanism of pushing robot	d^{pr}	
Rotation angle of grasping mechanism of pushing robot	ϕ^{pr}	
Estimated position of logistics cart	$\hat{y}^{lc}$	
Estimated orientation of logistics cart	$\cos\hat{\theta}^{lc}, \sin\hat{\theta}^{lc}$	
Time derivatives of above variables	15 dimensions	
Logistics cart size	L^{lc}	✓
Curvature	ρ	✓
Action before one step	$v^{sr}, w^{sr}, v^{pr}, w^{pr}$	
Action variables		
Each robot's velocity and angular velocity	$v^{sr}, w^{sr}, v^{pr}, w^{pr}$	

3.3. Reward shaping

The reinforcement learning controller is required to make the logistics cart follow the trajectory as accurately and speedily as possible. Therefore, the reward is shaped by three elements: a) logistics cart's position and orientation errors from the trajectory, b) logistics cart's velocity, and c) each robot's position and orientation relative to the logistics cart's position and orientation. The reward element a) is based on [38,39] and b) is based on [39], which are studies on trajectory tracking using reinforcement learning. In the real world, the true position, orientation, and velocity of the logistics cart cannot be obtained, but in simulation, the reward calculator can refer to the ground truth and then calculate the reward by utilizing it.

3.3.1. Logistics cart's position and orientation errors

First, we introduce the reward for precision of tracking the trajectory. This is divided into two elements: the position deviation and the orientation deviation. The reward for position deviation r_{pd} is defined as

$$r_{\mathrm{pd}} = \exp\left(-k_{\mathrm{pd}}|y^{\mathrm{lc}}|\right), \tag{1}$$

where y^{lc} is the logistics cart's position in the local y-coordinate and k_{pd} is a coefficient set to 7.5.

The reward for orientation deviation r_{od} is defined as

$$r_{\mathrm{od}} = \exp\left(-k_{\mathrm{od}}|\theta^{\mathrm{lc}}|\right), \tag{2}$$

where $|\theta^{\mathrm{lc}}|$ is the logistics cart's orientation in the local coordinates and k_{od} is a coefficient set to $13.5/\pi$.

3.3.2. Logistics cart's velocity

Ideally, the logistics cart should be transported as speedily as possible. Therefore, the reward for the logistics cart's velocity r_{dv} is defined as

$$r_{\mathrm{dv}} = k_{\mathrm{dv}} d_{\mathrm{d}}, \tag{3}$$

where d_{d} is the amount of the logistics cart's advancement along a given trajectory and k_{dv} is a coefficient set to 10. To calculate d_{d}, first, polar coordinates whose origin is the center of a circular trajectory are defined and the moving angle of the logistics cart in one step is defined as $\Delta\theta_{\mathrm{d}}$. Then, if the radius of curvature is R, d_{d} is calculated as $d_{\mathrm{d}} = R\Delta\theta_{\mathrm{d}}$.

3.3.3. Each robot's position and orientation relative to logistics cart's position and orientation

Stable holding of a logistics cart is important for users' sense of security. Thus, we introduce the reward for holding it as long as possible. For holding the logistics cart, each robot's position and orientation relative to the logistics cart's position and orientation are required to stay in a certain area. This area is decided by the range of motion of the touch mechanism d_{lim}, the width of the touch mechanism W, and the angle range of the rotation mechanism ϕ_{lim}. Therefore, the area is defined as

$$\begin{aligned} &x\text{-}axis : \frac{L^{\mathrm{lc}}}{2} + L^{\mathrm{gm}} - d_{\mathrm{lim}} \leq |r_x^{\{\mathrm{sr,pr}\}}| \leq \frac{L^{\mathrm{lc}}}{2} + L^{\mathrm{gm}}, \\ &y\text{-}axis : |r_y^{\{\mathrm{sr,pr}\}}| \leq \frac{W}{5}, \\ &orientation : |r_\theta^{\{\mathrm{sr,pr}\}}| \leq \phi_{\mathrm{lim}}, \end{aligned} \tag{4}$$

where $r_x^{\{\mathrm{sr,pr}\}}, r_y^{\{\mathrm{sr,pr}\}}, r_\theta^{\{\mathrm{sr,pr}\}}$ are each robot's position and orientation relative to the logistics cart's position and orientation and L^{gm} is the length from the robot's origin to the touch plate of the holding mechanism. L^{lc} is the length of the logistics cart (see Figure 3). The reward for relative position and orientation is defined as

$$r_{\mathrm{rp}} = \begin{cases} 1.0 & (both robots are in the area as defined by (4)) \\ 0.1 & (otherwise) \end{cases}. \tag{5}$$

Finally, the reward is defined by combining all elements:

$$r = \begin{cases} r_{\mathrm{dv}} r_{\mathrm{rp}} \left(r_{\mathrm{pd}} + r_{\mathrm{od}}\right) & (0 \leq r_{\mathrm{dv}}) \\ r_{\mathrm{dv}} \left(4 - r_{\mathrm{pd}} - r_{\mathrm{od}}\right) & (otherwise) \end{cases}. \tag{6}$$

This equation means that not only that the logistics cart tracks a given trajectory but also that an amount of the logistics cart advancement along a given trajectory is needed for obtaining the reward. If the logistics cart moves backwards along a given trajectory, the system is given a minus reward even if it tracks a given trajectory perfectly. We calculate the reward shaping by the product of the error elements and the velocity element as in [39]. This product makes each element improve for obtaining higher reward. As our research scenario allows a logistics cart to move backwards, we have added the reward for moving backwards.

3.4. Formation-based feedback controller

We utilize the feedback controller proposed in [28] for logistics cart transportation by formation-based control, which corresponds to the grasping strategy as the base controller. In this method, the system regards an object as a virtual leader (VL) and the robots as followers, and the leader-follower system controls the robots by feedback control for maintaining formation. We show later that the robots target orientation can be written simply.

The robots target is calculated as the relative state of VL. VL's target velocity, position, and orientation in local coordinates at time k are $\boldsymbol{v}_{\mathrm{tgt}}^{\mathrm{VL}}(k) = [v_{\mathrm{tgt}}^{\mathrm{VL}}, \omega_{\mathrm{tgt}}^{\mathrm{VL}}]^\top$, $\boldsymbol{x}_{\mathrm{tgt}}^{\mathrm{VL}}(k) = [0,0]^\top, \theta_{\mathrm{tgt}}^{\mathrm{VL}}(k) = 0$. Local coordinates are on a given trajectory and x-axis corresponds to the tangent of a given trajectory, so the logistics cart's target position and orientation are both zero. Also, VL's target position at time $k+1$ in local coordinates at time k is

$$\boldsymbol{x}_{\mathrm{tgt}}^{\mathrm{VL}}(k+1) = \begin{cases} \left[R\sin\omega_{\mathrm{tgt}}^{\mathrm{VL}} t_{\mathrm{s}}, \right. \\ \left. R(1-\cos\omega_{\mathrm{tgt}}^{\mathrm{VL}} t_{\mathrm{s}})\right]^\top & \left(\omega_{\mathrm{tgt}}^{\mathrm{VL}} \neq 0\right) \\ \left[v_{\mathrm{tgt}}^{\mathrm{VL}} t_{\mathrm{s}}, 0\right]^\top & \left(\omega_{\mathrm{tgt}}^{\mathrm{VL}} = 0\right) \end{cases}, \tag{7}$$

where R is the radius of the curvature of a given trajectory. In addition, t_{s} [s] is the time elapsed in one

step, which we set to 0.1 s in this paper. The supporting robot's target position at time k in local coordinates at time k is $\boldsymbol{x}^{\mathrm{sr}}_{\mathrm{tgt}}(k) = [L, 0]^\top$, and the supporting robot's target position at time $k+1$ in local coordinates at time k is written as $\boldsymbol{x}^{\mathrm{sr}}_{\mathrm{tgt}}(k+1) = \boldsymbol{x}^{\mathrm{VL}}_{\mathrm{tgt}}(k+1) + [L\cos\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}}, L\sin\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}}]^\top$.

The supporting robot's target velocity is $||\boldsymbol{x}^{\mathrm{sr}}_{\mathrm{tgt}}(k+1) - \boldsymbol{x}^{\mathrm{sr}}_{\mathrm{tgt}}(k)||/t_{\mathrm{s}}$. The result of calculating this equation is

$$v^{\mathrm{sr}}_{\mathrm{tgt}}(k) = \sqrt{v^{\mathrm{VL}2}_{\mathrm{tgt}} + 2L^2\left(1-\cos\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}}\right)/t_{\mathrm{s}}^2}. \quad (8)$$

Additionally, the supporting robot's target orientation in local coordinates at time k is calculated from its positions at times $k-1$ and k as

$$\begin{aligned}\theta^{\mathrm{sr}}_{\mathrm{tgt}}(k) &= \pi + \tan^{-1}\frac{y^{\mathrm{sr}}_{\mathrm{tgt}}(k) - y^{\mathrm{sr}}_{\mathrm{tgt}}(k-1)}{x^{\mathrm{sr}}_{\mathrm{tgt}}(k) - x^{\mathrm{sr}}_{\mathrm{tgt}}(k-1)}\\ &= \pi + \rho L - \frac{\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}}}{2}, \end{aligned} \quad (9)$$

where ρ is the curvature of a given trajectory, which means the target orientation can be calculated simply. The details of the calculation process are shown in the Appendix A. The first item π is added so that the supporting robot moves backwards. From the above, the supporting robot's targets in local coordinates at time k are

$$\begin{aligned}\boldsymbol{v}^{\mathrm{sr}}_{\mathrm{tgt}}(k) &= \left[v^{\mathrm{sr}}_{\mathrm{tgt}}, \omega^{\mathrm{sr}}_{\mathrm{tgt}}\right]^\top\\ &= \left[\sqrt{v^{\mathrm{VL}2}_{\mathrm{tgt}} + 2L^2\left(1-\cos\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}}\right)/t_{\mathrm{s}}^2}, \quad \omega^{\mathrm{VL}}_{\mathrm{tgt}}\right]^\top,\\ \boldsymbol{x}^{\mathrm{sr}}_{\mathrm{tgt}}(k) &= [L, 0]^\top,\\ \theta^{\mathrm{sr}}_{\mathrm{tgt}}(k) &= \pi + \rho L - \omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}}/2. \end{aligned} \quad (10)$$

The pushing robot's targets are calculated in the same way.

$$\begin{aligned}\boldsymbol{v}^{\mathrm{pr}}_{\mathrm{tgt}}(k) &= \left[v^{\mathrm{pr}}_{\mathrm{tgt}}, \omega^{\mathrm{pr}}_{\mathrm{tgt}}\right]^\top\\ &= \left[\sqrt{v^{\mathrm{VL}2}_{\mathrm{tgt}} + 2L^2\left(1-\cos\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}}\right)/t_{\mathrm{s}}^2},\right.\\ &\qquad \left.\omega^{\mathrm{VL}}_{\mathrm{tgt}}\right]^\top,\\ \boldsymbol{x}^{\mathrm{pr}}_{\mathrm{tgt}}(k) &= [-L, 0]^\top, \quad \theta^{\mathrm{pr}}_{\mathrm{tgt}}(k) = -\rho L - \omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}}/2. \end{aligned} \quad (11)$$

After calculating each robot's targets, each robot's position and orientation errors between the targets and the observations are calculated as

$$\begin{aligned}\boldsymbol{x}^{\{\mathrm{sr,pr}\}}_{e} &= \left[x^{\{\mathrm{sr,pr}\}}_{e}\ y^{\{\mathrm{sr,pr}\}}_{e}\right]^\top = {}^{\{\mathrm{sr,pr}\}}T_{\mathrm{trj}}\left(\boldsymbol{x}^{\{\mathrm{sr,pr}\}}_{\mathrm{tgt}} - \boldsymbol{x}^{\{\mathrm{sr,pr}\}}_{\mathrm{obs}}\right)\\ \theta^{\{\mathrm{sr,pr}\}}_{e} &= \theta^{\{\mathrm{sr,pr}\}}_{\mathrm{tgt}} - \theta^{\{\mathrm{sr,pr}\}}_{\mathrm{obs}}, \end{aligned} \quad (12)$$

where ${}^{\{\mathrm{sr,pr}\}}T_{\mathrm{trj}}$ is the transformation matrix from local coordinates to each robot's coordinates. $\boldsymbol{x}^{\{\mathrm{sr,pr}\}}_{\mathrm{obs}}, \theta^{\{\mathrm{sr,pr}\}}_{\mathrm{obs}}$ are the observation value of each robot's position and orientation. Following [40], each robot's velocity is calculated as

$$\begin{aligned}\pi^{\{\mathrm{sr,pr}\}}_{\mathrm{fb}}(\boldsymbol{o}_{\mathrm{fb}}) &= \begin{bmatrix} v^{\{\mathrm{sr,pr}\}}_{\mathrm{fb}} \\ \omega^{\{\mathrm{sr,pr}\}}_{\mathrm{fb}} \end{bmatrix}\\ &= \begin{bmatrix} v^{\{\mathrm{sr,pr}\}}_{\mathrm{tgt}}\cos\theta^{\{\mathrm{sr,pr}\}}_{e} + K_x x^{\{\mathrm{sr,pr}\}}_{e} \\ \omega^{\mathrm{VL}}_{\mathrm{tgt}} + v^{\{\mathrm{sr,pr}\}}_{\mathrm{tgt}} \\ \left(K_y y^{\{\mathrm{sr,pr}\}}_{e} + K_\theta \sin\theta^{\{\mathrm{sr,pr}\}}_{e}\right) \end{bmatrix}, \end{aligned} \quad (13)$$

where K_x, K_y, K_θ are the feedback gains and $\boldsymbol{o}_{\mathrm{fb}}$ is the observation for calculating the feedback controller output (refer to Table 1). In our work, the feedback gains are decided by the Tree-structured Parzen Estimator Approach (TPE) [41] (a Bayesian optimization method) so that the reward for one episode is maximized. For the implementation of TPE, we utilize Optuna [42]. This feedback controller is locally stable from the perspective of Lyapunov function, but it is not enough for transporting the logistics cart, as discussed later in Section 4.

3.5. Residual reinforcement learning

Here, we introduce our logistics cart transportation controller using residual reinforcement learning. In residual reinforcement learning, the control output is calculated as the sum of the base controller output and the reinforcement learning controller output. The output of the reinforcement learning controller is defined as each robot's linear velocity $v^{\{\mathrm{sr,pr}\}}_{\mathrm{res}}$ and rotation velocity $\omega^{\{\mathrm{sr,pr}\}}_{\mathrm{res}}$; thus, the reinforcement learning controller outputs four dimensions of action:

$$\pi_{\mathrm{res}}(\boldsymbol{o}_{\mathrm{fb}}, \boldsymbol{o}_{\mathrm{rl}}) \rightarrow \left[v^{\mathrm{sr}}_{\mathrm{res}},\ \omega^{\mathrm{sr}}_{\mathrm{res}},\ v^{\mathrm{pr}}_{\mathrm{res}},\ \omega^{\mathrm{pr}}_{\mathrm{res}}\right]^\top, \quad (14)$$

where $\boldsymbol{o}_{\mathrm{rl}}$ is the observation that is used for reinforcement learning. Finally, each robot's control output is calculated as

$$\begin{aligned}\pi(\boldsymbol{o}_{\mathrm{fb}}, \boldsymbol{o}_{\mathrm{rl}}) &= \left[v^{\mathrm{sr}},\ \omega^{\mathrm{sr}},\ v^{\mathrm{pr}},\ \omega^{\mathrm{pr}}\right]^\top\\ &= \pi_{\mathrm{fb}}(\boldsymbol{o}_{\mathrm{fb}}) + \pi_{\mathrm{res}}(\boldsymbol{o}_{\mathrm{fb}}, \boldsymbol{o}_{\mathrm{rl}}), \end{aligned} \quad (15)$$

where $\pi_{\mathrm{fb}}(\boldsymbol{o}_{\mathrm{fb}}) = [\pi^{\mathrm{sr}}_{\mathrm{fb}}(\boldsymbol{o}_{\mathrm{fb}}), \pi^{\mathrm{pr}}_{\mathrm{fb}}(\boldsymbol{o}_{\mathrm{fb}})]^\top$.

3.6. Learning algorithm

We use Twin Delayed DDPG (TD3) [6] as the learning algorithm. We considered using another promising method, Soft-Actor Critic [7], which utilizes the stochastic action and maximizes the trade-off objective between the expected sum of rewards and the entropy of the stochastic action. However, in residual reinforcement learning, we think that exploitation is more important than exploration, so we adopt TD3, which is exploration by fixed distributed noise.

The reinforcement learning objective is to maximize the expected reward, defined as

$$J(\phi) = \mathbb{E}_{s_i \sim p_\pi, a_i \sim \pi}\left[\sum_{i=0}^{T} \gamma^i r_i\right], \tag{16}$$

where π is the policy, ϕ is the parameters in the policy, and γ is the time discount factor. Also, the state-action value function Q is defined as

$$Q_\theta^\pi(s_t, a_t) = \mathbb{E}_{s_i \sim p_\pi, a_i \sim \pi}\left[\sum_{i=t}^{T} \gamma^i r_i | s_t, a_t\right], \tag{17}$$

which means that the action a_t causes the time-discounted and cumulated reward when the state is s_t. Q-learning minimizes the temporal differential error (TD error). Q-function is written as follows based on a Bellman equation:

$$Q_\theta^\pi(s_t, a_t) = r_t + \gamma \mathbb{E}_{s_{t+1}, a_{t+1}}\left[Q_\theta^\pi(s_{t+1}, a_{t+1})\right]. \tag{18}$$

We utilize multi-step learning, which is a method for stabilizing the learning process by using the forward-view n-step rewards. In Q-learning, the parameters of the target network are defined as $\bar{\theta}$ and the target value is defined as

$$y = r_t + \gamma r_{t+1} + \cdots + \gamma^n Q_{\bar{\theta}}^\pi(s_{t+n}, a_{t+n}). \tag{19}$$

Therefore, the objective for minimizing the TD error is written as

$$L_Q(\theta) = \frac{1}{N}\sum_i \text{HuberLoss}\left(y - Q_\theta^\pi(s_i, a_i)\right),$$
$$\text{HuberLoss}(\delta) = \begin{cases} \delta^2/2 & (|\delta| < 1) \\ |\delta| - 0.5 & (otherwise) \end{cases}, \tag{20}$$

where Huber loss is used for robustness to outliers. Also, $\bar{\theta}$, which is the parameter of the target network, is updated for tracking θ. Concretely, $\bar{\theta}$ is updated as $\bar{\theta} \leftarrow (1-\tau)\bar{\theta} + \tau\theta$, where τ is the hyper-parameter and the smaller τ is, the more slowly the target network is updated. Additionally, TD3 utilizes two methods for Q-learning: (i) clipped double Q-learning, where two Q-functions are used and a small value of the two Q-functions is adopted as the target value for avoiding overestimation, and (ii) target policy smoothing, where noise is added to the action when the Q value of the target network in the target value calculation is calculated for stabilizing learning.

Then, for optimizing the policy, we utilize the deterministic policy gradient method proposed by Silver et al. [43]. The gradient is calculated for maximizing $J(\phi)$, so it is obtained by using the Q-function as follows:

$$\nabla_\phi J(\phi) = \mathbb{E}_{s \sim p_\pi}\left[\nabla_a Q(s,a)|_{a=\pi(\cdot|s)} \nabla_\phi \pi_\phi(\cdot|s)\right]. \tag{21}$$

The basic policy gradient method updates the Q-function and the policy in one learning step, but TD3 makes the policy update once every multiple steps of learning, which is called a delayed policy update. In addition, we utilize prioritized experience replay [44] and distributed learning for learning efficiently.

Prioritized experience replay is a method where the experience that has a large TD error is sampled with priority, which makes it possible to learn efficiently. The priority is defined as

$$p_i = \left|y - Q_\theta^\pi(s_i, a_i)\right| + \epsilon, \tag{22}$$

where ϵ is a small value for avoiding the priority becoming zero. Then, the probability of the i-th experience is

$$P(i) = \frac{p_i^\alpha}{\sum_i p_i^\alpha}, \tag{23}$$

where α is the hyper-parameter. Also, importance sampling is applied for compensating the biased sampling. The weight for the importance sampling is defined as

$$w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)}\right)^\beta, \tag{24}$$

where β is the hyper-parameter for deciding the degree of importance sampling and its bias is corrected if $\beta = 1$. N is the size of replay memory. Also, for stabilizing, the weight is normalized to $1/\max_i w_i$. By using Equation (23), the weight is written simply as

$$w_i \leftarrow \frac{w_i}{w_{\max}} = \left(\frac{1}{N} \cdot \frac{\sum_i p_i^\alpha}{p_i^\alpha}\right)^\beta \left(\frac{1}{N} \cdot \frac{\sum_i p_i^\alpha}{p_{\min}^\alpha}\right)^{-\beta} = \left(\frac{p_{\min}}{p_i}\right)^{\alpha\beta}. \tag{25}$$

For distributed learning, eight actor modules and one learner module are executed in parallel. The algorithm using prioritized experience replay and distributed learning is similar to the one in [14]. In [14], a large number

of actors (~ 256) is executed in parallel, so in the actor module, the priority is calculated. However, our implementation does not calculate the priority in the actor module because paralleled actor modules are not large. The pseudo codes are shown in Algorithms 1 and 2.

Algorithm 1 Actor.

Input: index of Actor i, Learner, policy update interval K, multi-step learning size n,
standard deviation of action noise σ, time discount factor γ, feedback policy π_{fb}:

Initialize policy network π_ϕ
Initialize Environment env
Initialize local replay memory LR
for $episode = 0, 1, \ldots$ **do**
  env.reset(i)
  $s_o \leftarrow$ env.observation()
  for $t = 0, 1, \ldots$ **do**
    if $tmodK = 0$ **then** $\pi_\phi \leftarrow$ *Learner.copy_policy*() **end if**
    $a_t \leftarrow \pi_\phi(s_t) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma)$
    $a'_t = \pi_{\mathrm{fb}}(s_t) + a_t$
    $r_t, done = env.step(a'_t)$
    $s_{t+1} \leftarrow$ env.observation()
    LR.add$(s_t, a_t, r_t, s_{t+1}, done)$
    if LR.size() $\geq n$ or done **then**
      $p \leftarrow Learner.p_{max}$
      $(s_t, a_t, \sum_{\tau=0}^{n} \gamma^\tau r_{t+\tau-1}, s_{t+n}, done) \leftarrow$ LR.make_n_step_return()
      Learner.add_to_replay_memory(p, $(s_t, a_t, \sum_{\tau=0}^{n} \gamma^\tau r_{t+\tau-1}, s_{t+n}, done)$)
    end if
    if done **then** break **end if**
  end for
end for

Algorithm 2 Learner.

Input: standard deviation of target policy smoothing noise $\bar{\sigma}$, noise clipping coefficient c,
time discount factor γ, multi-step return size n,target network update coefficient τ,
policy update frequency F, mini-batch size N:
Output: optimal policy π^*

Initialize critic networks $Q_{\theta_1}, Q_{\theta_2}$ and policy network π_ϕ
Initialize target networks $\bar{\theta}_1, \bar{\theta}_2, \bar{\phi} \leftarrow \theta_1, \theta_2, \phi$
Initialize ReplayMemory R
$p_{max} \leftarrow 1.0$
for $i = 0, 1, \ldots$ **do**
  mini-batch$(p, w, (s, a, r, s', done)) \leftarrow$ R.prioritized_sample()
  $a' \leftarrow \pi_{\bar{\phi}}(\cdot|s) + \epsilon, \quad \epsilon \sim \mathrm{clip}(\mathcal{N}(0, \tilde{\sigma}), -\mathrm{c}, \mathrm{c})$
  $y \leftarrow r + \gamma^n \min_{j=1,2} Q_{\bar{\theta}_j}(s', a')$
  update p by TD error
  if $p > p_{max}$ **then** $p_{max} \leftarrow p$ **end if**
  update critics θ_j by $L_Q(\theta_j) = N^{-1} \sum w\mathrm{HuberLoss}(y, Q_{\theta_j}(s, a))$
  if $imodF = 0$ **then**
    update policy ϕ by $J_\pi(\phi) = N^{-1} \sum Q_{\theta_1}(s, \pi_\phi(\cdot|s))$
    update target networks:
    $\bar{\theta}_j \leftarrow \tau\bar{\theta}_j + (1-\tau)\theta_j$
    $\bar{\phi} \leftarrow \tau\bar{\phi} + (1-\tau)\phi$
  end if
end for

4. Experiments

4.1. Simulation setting

We utilize Pybullet [9] for constructing the simulation environment. Three logistics cart sizes $\{0.6, 0.795, 0.8\}$ m are prepared. These models and the robot model, which are based on real-world logistics carts and robots, are shown in Figure 4. The weight of the load is sampled from a uniform distribution between 120 and 130 kg at the start of the episode if the size of the logistics cart is $\{0.795, 0.8\}$ m or between 100 and 110 kg if the size of the logistics cart is 0.6 m. The range of the curvature is $[-0.06, 0.66]$ m^{-1} and this is divided into eight areas uniformly. Each area is allocated to one of the actor modules. The value of the curvature is sampled from a uniform distribution in the allocated area at the start of the episode. The frequency of the observation and control is 10 Hz. The termination conditions of the episode are that 30 s has passed or one or more of the following conditions have persisted for 1 s.

- The deviation from the trajectory is larger than 0.4 m
- The orientation deviation from the reference is larger than $\pi/4.5$ rad
- Moving distance along the trajectory is smaller than 0.005 m
- At least one of the robots has been out of a certain area (refer to Equation (4))

If the episode satisfies at least one of the termination conditions (except that time is up), the reinforcement learning controller receives a reward of -10.

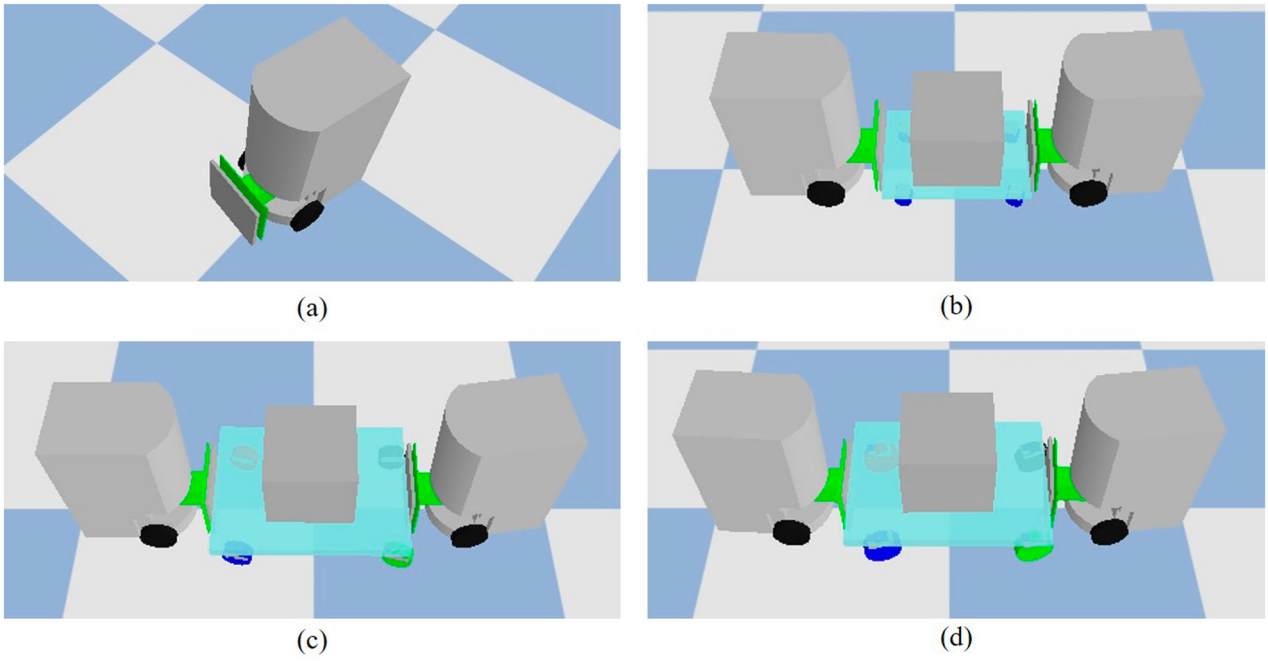

Figure 4. (a) Robot model. (b) Logistics cart model (size: 0.6 m). (c) Logistics cart model (size: 0.795 m). (d) Logistics cart model (size: 0.8 m).

4.2. Implementation details of proposed methodology

The output of the feedback controller is calculated by Equation (13). The target velocity of the feedback controller $v_{\mathrm{tgt}}^{\mathrm{VL}}$ is set to 0.5 m/s. The feedback gains are set to the same values as the feedback controller in Section 4.3. The policy network and the Q network architectures are shown in Figure 5. In training, we use the Adam [45] optimizer, and hyper-parameters are set as follows: learning rate of 3×10^{-4}, replay memory size of 10^5, target network update coefficient τ of 0.005, time discount factor γ of 0.99, policy update interval K of 100, multi-step learning size n of 4, batch size of 256, standard deviation of action noise σ of $0.1 \times (a_{\max} - a_{\min})$, standard deviation of target policy smoothing noise $\bar{\sigma}$ of $0.2 \times (a_{\max} - a_{\min})$, noise clipping coefficient c of $0.5 \times (a_{\max} - a_{\min})$, policy update frequency F of 2, and the hyper-parameters of prioritized experience replay α, ϵ, β of 0.6, 0.01, 0.4.

4.3. Baselines

4.3.1. RL-only

RL-only has each robot's linear velocity and angular velocity, both of which are four dimensions of continuous action. The network architecture and hyper-parameters are the same as the residual reinforcement learning implementation. RL-only does not use knowledge and instead learns from scratch.

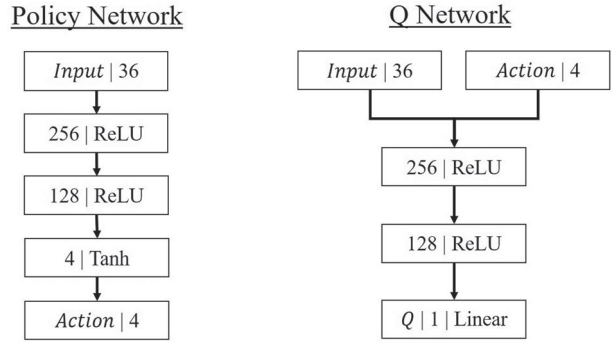

Figure 5. Network architecture.

4.3.2. Feedback controller

The output of the feedback controller is calculated by Equation (13). The target velocity of the logistics cart is set to 0.5 m/s and the target angular velocity of the logistics cart is set to $\omega_{\mathrm{tgt}}^{\mathrm{VL}} = \rho v_{\mathrm{tgt}}^{\mathrm{VL}} = 0.5\rho$, where ρ is the curvature. The feedback gains K_x, K_y, K_θ are explored in the range $[0, 10]$ by TPE with 100 steps and are set to values that maximize the expected total reward of one episode. The expected total reward for one episode is calculated by running 15 episodes that are a combination of three logistics cart sizes $\{0.6, 0.795, 0.8\}$ m and five curvatures $\{0, 0.15, 0.3, 0.45, 0.6\}$ m^{-1}. The weight of the load is set to 120 kg at the start of the episode if the size of the logistics cart is $\{0.795, 0.8\}$ m or to 100 kg if the size of the logistics cart is 0.6 m. As a result, the feedback gains are

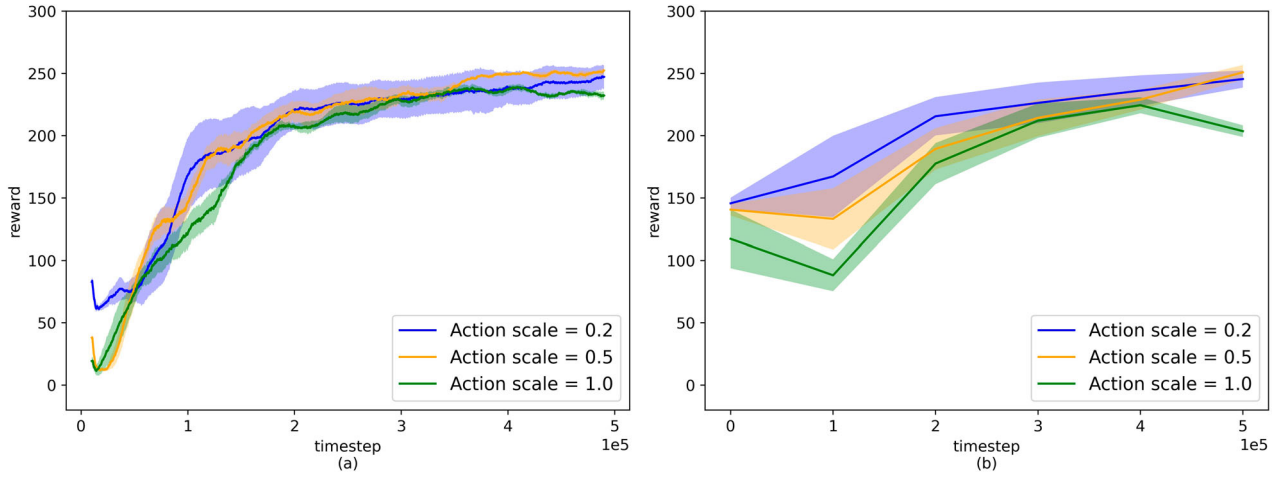

Figure 6. Learning curves of action scales' residual RL (a) with exploration noise and (b) without exploration noise.

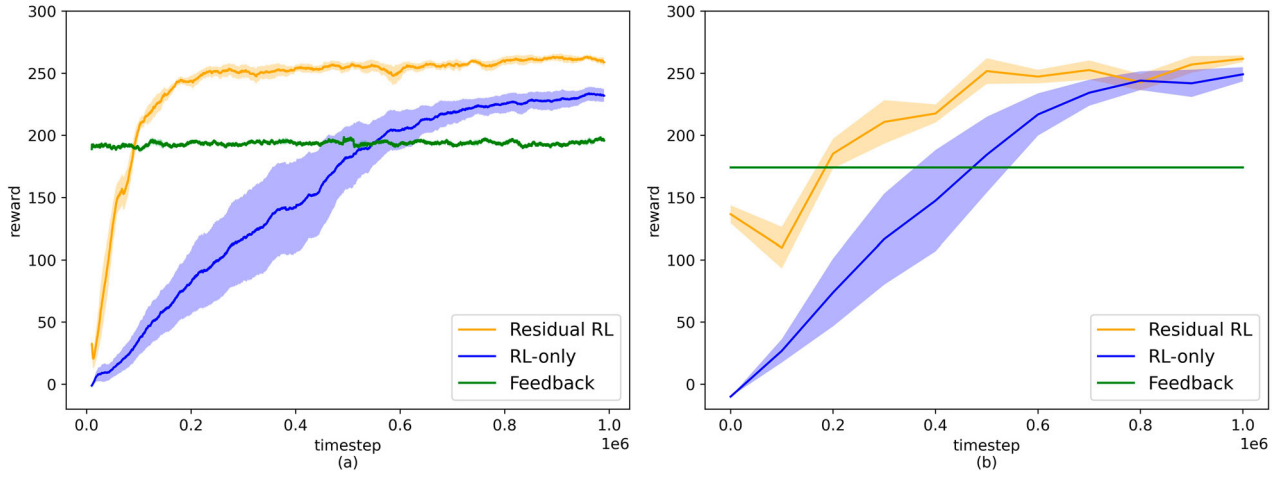

Figure 7. Learning curve of residual RL, RL-only, and feedback controller (a) with exploration noise and (b) without exploration noise.

set to $(K_x, K_y, K_\theta) = (3.82, 6.02, 1.43)$ and these are fixed during evaluation.

4.4. Learning evaluation

4.4.1. Effect of residual RL's action scale

Three action scales of residual reinforcement learning are compared. When the action scale is 1.0, the linear velocity range of residual reinforcement learning is $[-1.0, 1.0]$ m/s and its rotation velocity is $[-\pi/3, \pi/3]$ rad/s. The results of five trials of each action scale are shown in Figure 6. The left side of the figure plots the smoothed reward during training. The right side plots the results evaluated by executing 15 episodes that are a combination of three logistics cart sizes $\{0.6, 0.795, 0.8\}$ m and five curvatures $\{0, 0.15, 0.3, 0.45, 0.6\}$ m^{-1} with models that are stored for each of 10^5 experiences. The rule for sampling the weight of the load is the same as in Section 4.3.2. The results on the left are evaluated with exploration noise and those on the right without exploration noise. The lines mean the average and the shaded region represents half the standard deviation.

From Figure 6, we can see that the large action scale delays improvement of the policy. The maximum reward without exploration noise is obtained when the action scale is 0.5 and the timestep is 5×10^5; therefore, the action scale is set to 0.5.

4.4.2. Feedback controller vs RL-only vs residual RL

Figure 7 shows the learning curves. As in Section 4.4.1, the left side of the figure shows the training results and the right side shows the results of running episodes without exploration noise. The results of the feedback controller are also plotted, where those on the right side do not have a shaded area because Pybullet is the deterministic simulator.

Residual reinforcement learning achieves more efficient learning and a smaller standard deviation than

reinforcement learning. Final reward of residual reinforcement learning is slightly higher than that of reinforcement learning. Also, from the results on the right, it is clear that the reinforcement learning controllers acquire a higher reward than the feedback controller.

4.5. Simulation experiments

Next, we compare the reinforcement learning controllers and the feedback controller in a simulation environment. For this evaluation, each model of the reinforcement learning controllers at 5×10^5 steps is utilized. The expected total reward for one episode is calculated by running 15 episodes that are a combination of three logistics cart sizes $\{0.6, 0.795, 0.8\}$ m and five curvatures $\{0, 0.15, 0.3, 0.45, 0.6\}$ m^{-1}. The weight of the load is set to 120 kg at the start of the episode if the size of the logistics cart is $\{0.795, 0.8\}$ m or to 100 kg if the size of the logistics cart is 0.6 m. These conditions are the same as in Section 4.3.2.

4.5.1. Metrics

We evaluate the simulation results with the following five metrics. **Reward** is the total reward for one episode calculated by Equation (6). **Average velocity** is the average velocity of the logistics cart for one episode. **Trajectory error** is the average deviation from the trajectory for one episode, which is defined as the shortest distance between the logistics cart and the trajectory. **Orientation error** is the average deviation from the reference orientation that is the tangential direction of the trajectory for one episode. **Holding ratio** is the average ratio of holding the logistics cart with the robots for one episode. If the robots stay in area defined by Equation (4), holding is considered to be maintained.

4.5.2. Simulation results

The results of the simulation experiments are shown in Table 2. Metrics without rewards are set to '–' if the reward is smaller than 0, as metrics without rewards cannot be evaluated precisely when the logistics cart is not transported far enough. The results of the learning-based controllers are represented by the average values of five learned models.

The feedback controller cannot make the logistics cart track the trajectory when the curvature is large, so its reward is lower than that of the learning-based controller. Also, the residual reinforcement learning controller improves the transportation performance compared to the feedback controller in many conditions. The reinforcement learning controller only performed better than the other methods in the Holding ratio.

Figures 8 and 9 show snapshots of the simulation experiments. The feedback controller cannot keep holding the logistics cart, while in contrast, the residual reinforcement learning controller keeps holding it and achieves stable transportation.

Figure 10 shows the residual reinforcement learning controller's output in simulation experiments. The amount of compensation of linear velocity is large in the beginning of the movement. That of angular velocity is large throughout the episode.

4.6. Real-world experiments

Next, in real-world experiments, we compare the feedback controller and the residual reinforcement learning controller that obtained a higher reward than RL-only controller. We use the residual reinforcement learning controller learned in the simulation environment with no additional parameter tuning.

We run the experiment twice for each of 15 conditions that are a combination of three logistics cart sizes $\{0.6, 0.795, 0.8\}$ m and five curvatures $\{0, 0.15, 0.3, 0.45, 0.6\}$ m^{-1}. The weight of the load is set to 120 kg if the size of the logistics cart is $\{0.795, 0.8\}$ m or to 100 kg if the size of the logistics cart is 0.6 m. Also, the error rate of relative positions between robots by camera on ceiling is up to 1%, which is evaluated by a total station as a measurement instrument.

4.6.1. Real-world results

The results of the real-world experiments are shown in Table 3. The residual reinforcement learning controller has a better performance for the logistics cart transportation than the feedback controller, the same as in the simulation experiments. A snapshot of the real-world experiment is shown in Figure 11. In this figure, when the curvature is 0.6 m^{-1}, the feedback controller cannot transport the logistics cart because the holding mechanism becomes unfastened. This behavior is similar to the results of the simulation experiments shown in Figure 11. In contrast, the residual reinforcement learning controller can keep holding the logistics cart and achieves stable transportation.

5. Discussion

5.1. Effect of residual RL's action scale

The larger the action scale becomes, the worse the learning efficiency gets, as the high cost of exploration makes the action space large. In contrast, a small action scale reduces the better local minima in the action space, so the reward may be small. The fact that the reward becomes

Table 2. Results of simulation experiments by the controllers at 5×10^5 steps. Feedback: Feedback controller, RL: RL-only controller, RRL: Residual RL controller.

		Reward			Average velocity [m/s]			Trajectory error [m]			Orientation error [rad]			Holding ratio		
Logistics cart size [m]	Curvature	Feedback	RL	RRL	Feedback	RL	RRL	Feedback	RL	RRL	Feedback	RL	RRL	Feedback	RL	RRL
0.6	0	**284.70**	203.19	264.12	**0.491**	0.404	**0.491**	**0.0039**	0.0324	0.0089	**0.0032**	0.0341	0.0299	0.977	0.979	**0.980**
	0.15	**278.65**	214.51	269.41	0.487	0.427	**0.488**	**0.0046**	0.0405	0.0085	**0.0098**	0.0323	0.0176	0.977	**0.993**	0.982
	0.30	257.99	198.49	**271.61**	0.480	0.436	**0.482**	0.0108	0.0543	**0.0067**	0.0288	0.0630	**0.0148**	0.977	**0.996**	0.993
	0.45	**252.2**	181.61	211.27	**0.470**	0.423	0.449	**0.0100**	0.0461	0.0167	0.0337	0.1008	**0.0202**	0.977	**0.993**	0.888
	0.60	−9.61	164.61	**251.02**	–	0.399	**0.456**	–	0.0477	**0.0118**	–	0.1203	**0.0187**	–	0.975	**0.999**
0.795	0	**282.05**	211.41	264.88	0.492	0.402	**0.496**	**0.0028**	0.0275	0.0158	**0.0096**	0.0237	0.0188	0.973	**0.992**	0.976
	0.15	**273.60**	213.04	269.20	0.489	0.423	**0.491**	**0.0070**	0.0455	0.0094	**0.0153**	0.0330	0.0206	0.977	**0.996**	0.986
	0.30	242.86	194.21	**262.08**	0.470	0.428	**0.480**	0.0194	0.0596	**0.0107**	0.0333	0.0478	**0.0222**	0.977	**0.993**	0.992
	0.45	−9.60	169.65	**251.80**	–	0.398	**0.465**	–	0.0289	**0.0145**	–	0.0502	**0.0229**	–	0.916	**0.997**
	0.60	−9.61	125.78	**240.78**	–	0.385	**0.447**	–	0.0820	**0.0131**	–	0.1010	**0.0291**	–	0.925	**1.000**
0.8	0	**276.67**	212.79	211.16	**0.488**	0.398	0.449	**0.0045**	0.0237	0.0111	**0.0127**	0.0214	0.0141	0.973	**0.991**	0.777
	0.15	265.06	207.35	**266.35**	**0.486**	0.417	**0.486**	0.0138	0.0474	**0.0091**	**0.0157**	0.0377	0.0206	0.977	**0.997**	0.979
	0.30	241.28	203.45	**258.57**	0.469	0.422	**0.475**	0.0186	0.0617	**0.0119**	0.0354	0.0421	**0.0217**	0.97	**0.989**	0.988
	0.45	−7.34	177.34	**248.44**	–	0.406	**0.460**	–	0.0472	**0.0151**	–	0.0882	**0.0237**	–	0.932	**0.993**
	0.60	−9.66	89.45	**235.47**	–	0.360	**0.440**	–	0.1055	**0.0129**	–	0.1508	**0.0333**	–	0.950	**0.995**

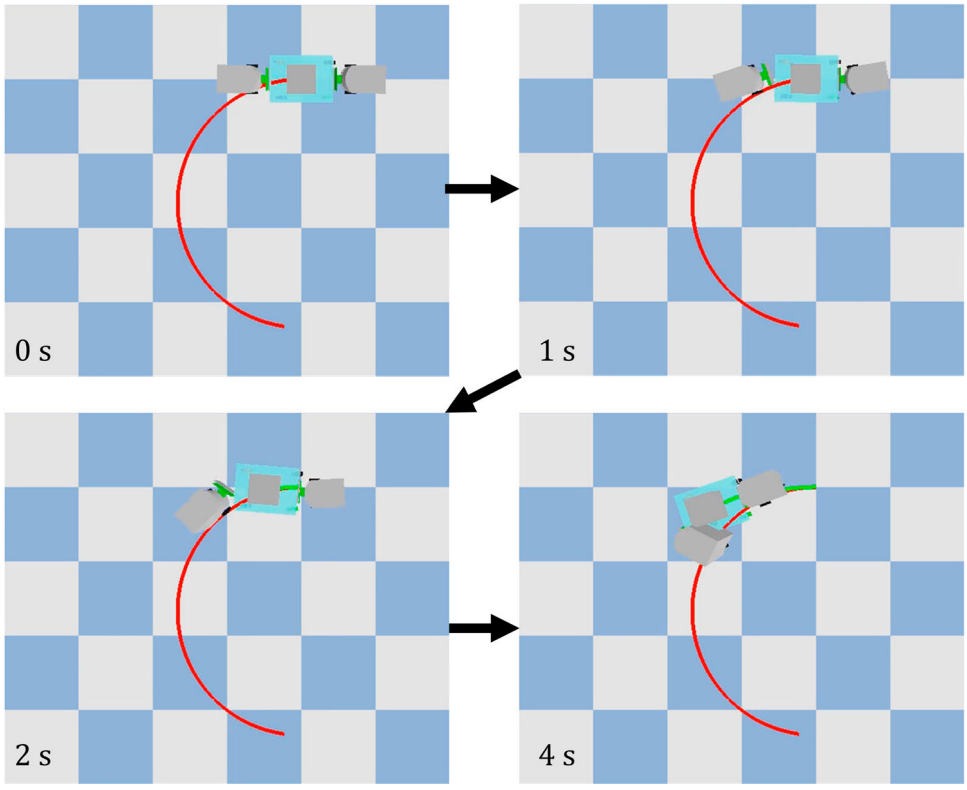

Figure 8. Example of feedback controller simulation experiment results. Logistics cart size is 0.8 m and trajectory curvature is 0.6 m^{-1}. Red line is the given trajectory and green line is actual trajectory of the logistics cart.

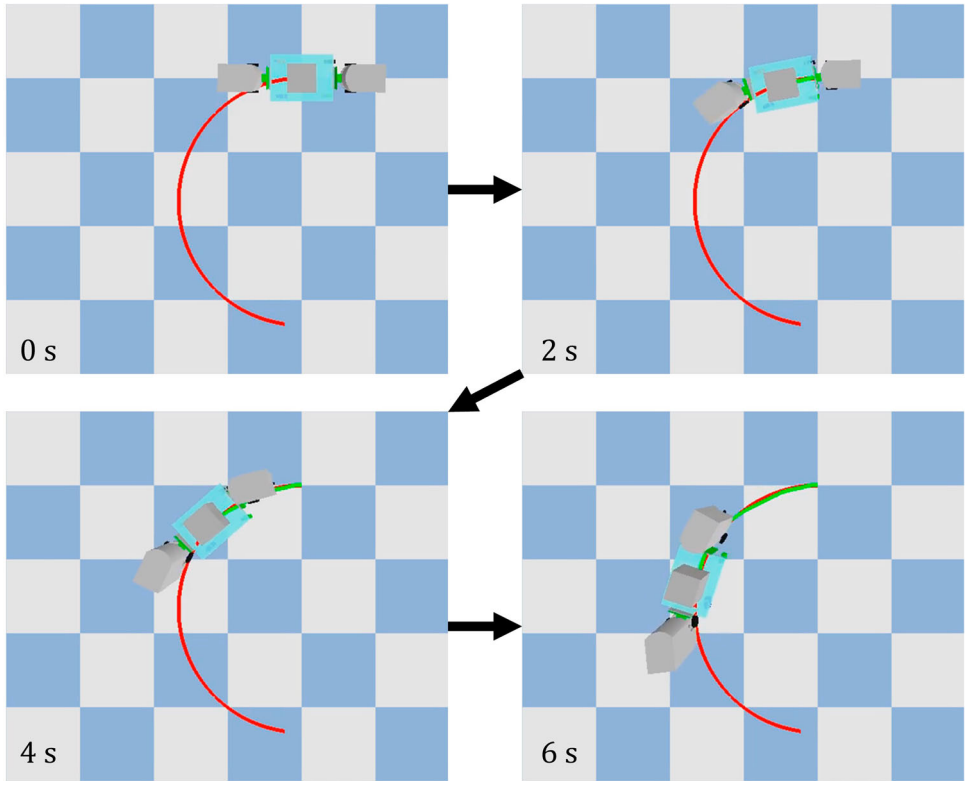

Figure 9. Example of residual RL controller simulation experiment results. Logistics cart size is 0.8 m and trajectory curvature is 0.6 m^{-1}. Red line is the given trajectory and green line is actual trajectory of the logistics cart.

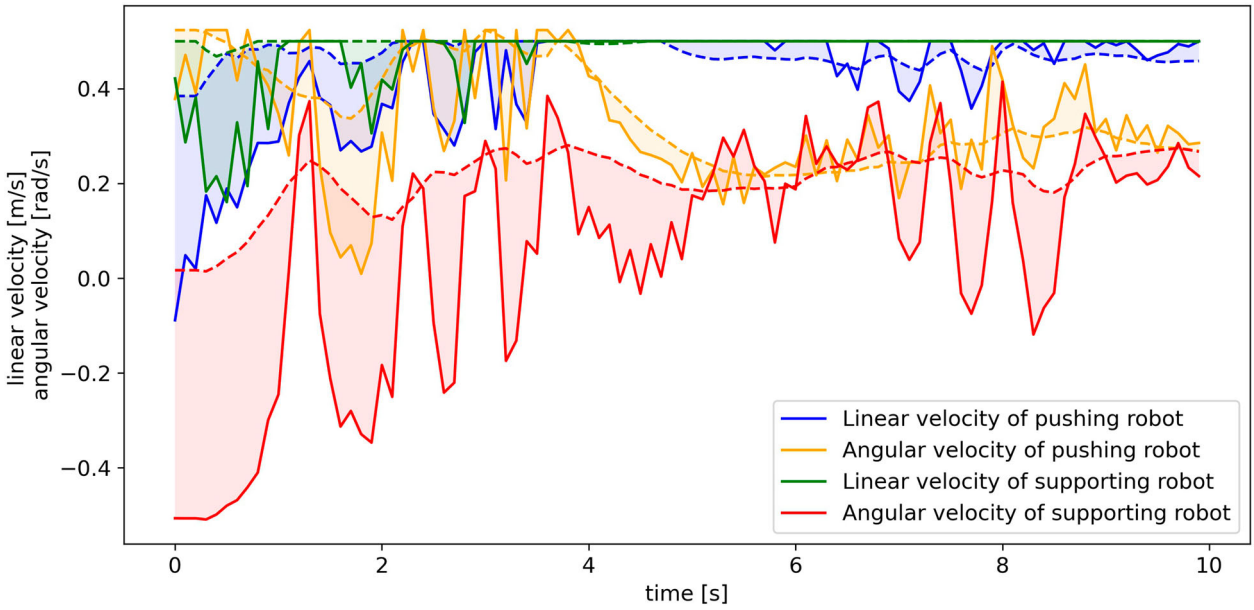

Figure 10. History of the residual RL controller's output for 10 s in the simulation experiment, where logistics cart size is 0.8 m and trajectory curvature is 0.6 m^{-1}. Dashed lines mean the output of the feedback controller and shaded regions mean the amount of the compensation by residual RL.

Table 3. Results of real-world experiments: (the number of successes) / (the number of experiments).

		Curvature [/m]				
Method	Logistics cart size [m]	0	0.15	0.30	0.45	0.60
Feedback controller	0.6	2/2	2/2	2/2	2/2	0/2
	0.795	2/2	2/2	2/2	1/2	0/2
	0.8	2/2	2/2	2/2	1/2	0/2
Residual RL controller	0.6	2/2	2/2	2/2	2/2	**2/2**
	0.795	2/2	2/2	2/2	**2/2**	**2/2**
	0.8	2/2	2/2	2/2	**2/2**	**2/2**

small if the action scale is 0 reinforces this consideration. Thus, the appropriate action scale has the potential for accelerating learning speed. In our work, we consider that robots basically do not need to move backward, so we can adopt a small action scale. As a result, the action scales 0.2 and 0.5 acquire the highest reward, with 0.5 obtaining a slightly higher one.

5.2. Learning curve of residual RL and RL-only

Residual reinforcement learning has a better learning efficiency than reinforcement learning. One reason for this result is that utilizing the feedback controller as a base controller prompts the gathering of more varied experiences. Also, residual reinforcement learning has the smaller standard deviation than reinforcement learning. This means that residual reinforcement learning is stable learning method. The reason of this result is that it is hard to stagnate on an unfavorable local optimum because of base controller. These advantages of residual reinforcement learning are important because they accelerate learning speed and reduce the number of trials.

From Figure 7, in the early period of learning, the residual reinforcement learning controller's reward is lower than that of the feedback controller. This is presumably because the Q-function does not acquire an accurate model, so it cannot evaluate the controller precisely and the controller cannot be improved. One idea for coping with this problem is that initial output of residual is set to 0. However, the results in previous research [17] indicates that this idea does not avoid the deterioration of the controller. We tried this idea in our environment, although it did not avoid this deterioration. The method to avoid this deterioration is future work.

5.3. Simulation experiments

Table 2 shows that the residual reinforcement learning controller has a better performance than both the reinforcement learning controller and the feedback controller under many conditions. In particular, when the curvature is large, the feedback controller cannot transport the logistics cart, while the residual and reinforcement learning controllers can.

The reason the learning-based controller is better than the feedback controller is that it can utilize information that is not used by the feedback controller and its output is nonlinearized. The feedback controller utilizes only each robot's position and orientation, whereas the

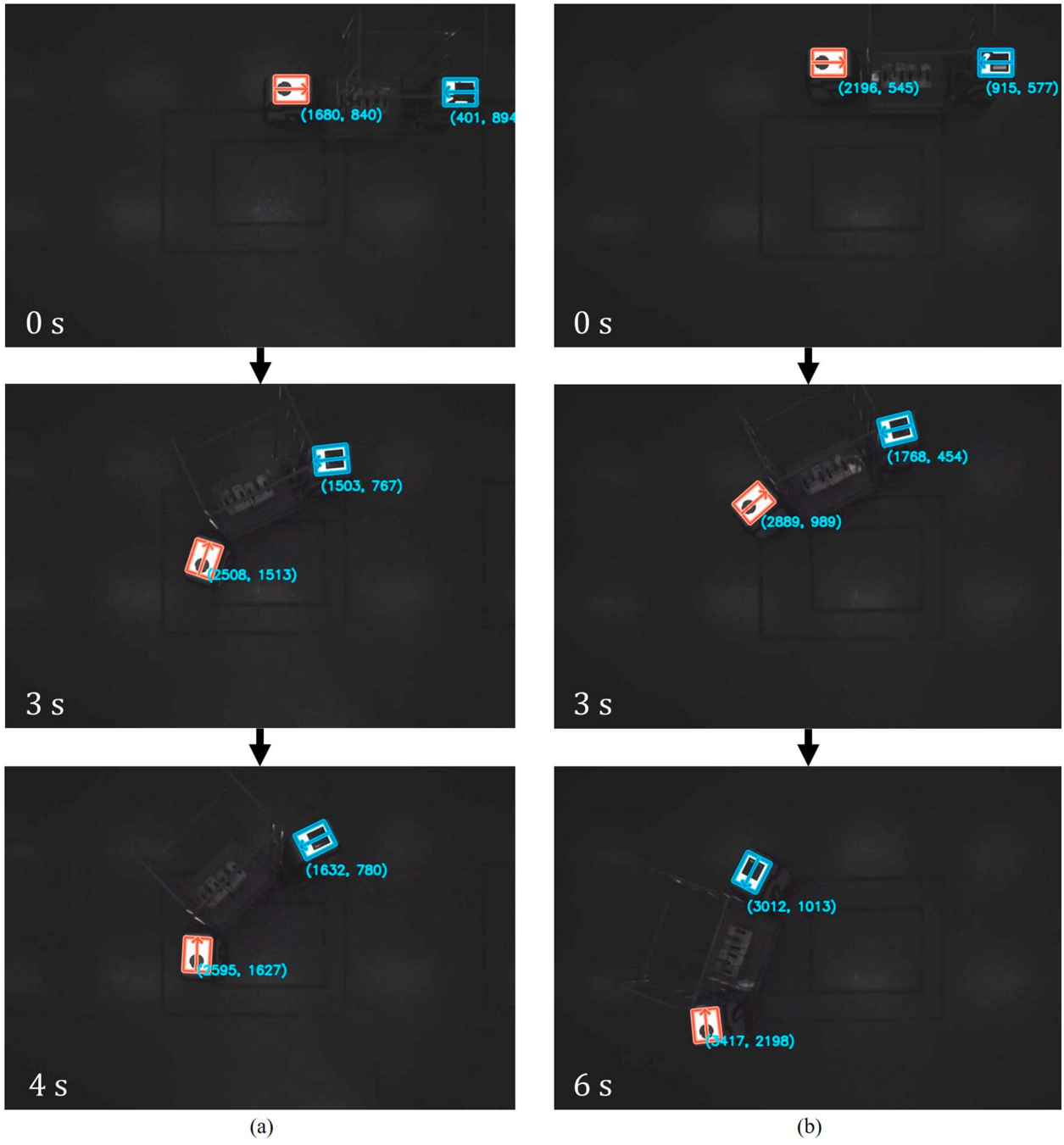

Figure 11. Example of real-world experiment results. (a) Feedback controller, where logistics cart size is 0.8 m and trajectory curvature is 0.6 m^{-1}. (b) Residual RL controller, where logistics cart size is 0.8 m and trajectory curvature is 0.6 m^{-1}.

learning-based controller utilizes the additional information shown in Table 1.

On the other hand, in some cases with a small curvature's trajectory, the feedback controller has a better performance than the residual reinforcement learning controller. This means that the residual reinforcement learning controller deteriorates the policy when it acquires the control law for transporting the logistics cart with a large curvature's trajectory. Avoiding this deterioration and creating a controller that has a better performance than the feedback controller in all available states remain issues for future work.

From Figure 8, the logistics cart cannot track the trajectory and so orientation error is accumulated. Then, when the robots try to modify this error, they cannot keep holding the logistics cart. This occurs because the feedback controller controls the robots' state and does not consider the logistics cart's state. Another reason for this

result is that the feedback controller makes only current errors become 0. In contrast, the residual reinforcement learning controller learns the policy for reducing the errors of the logistics cart and considers the long-period accumulated reward, so it can keep holding the logistics cart, as shown in Figure 9.

Figure 10 shows that residual reinforcement learning compensates the output of the feedback controller. The reason the amount of compensation of angular velocity is large is that the residual helps to change the orientation of the logistics cart for making it track the trajectory.

5.4. Real-world experiments

The results of the real-world experiments show that the residual reinforcement learning controller learned in simulation environments can be transferred to real-world control. However, the results of the real-world experiments differ slightly from the results of the simulation experiments. This difference presumably stems from some gap between the simulation and the real world, e.g. the friction of the wheels of the logistics cart, the gap of the holding position of the robots, and/or observation noise. The results of the real-world experiments suggest that the residual reinforcement learning controller absorbs the gap between the simulation and the real world. Randomization may be one cause of this, e.g. exploration noise and the noise added to the load weight. While reducing the gap between the simulation and the real world was not considered in our work, it is important to clarify how to transfer the reinforcement learning controller learned in a simulation environment to real-world tasks more robustly.

6. Conclusion

We proposed a system for logistics cart transportation with a residual reinforcement learning controller. The proposed controller is more sample efficient than a reinforcement learning controller trained from scratch and has a higher performance than the feedback controller. We showed that using simulation reduces the cost of gathering experiences, and the results of real-world experiments suggest that the residual reinforcement learning controller learned in a simulation environment can be transferred to real-world control.

As future work, we will investigate how to make the controller higher performance than the feedback controller in all available states and reduce the difference between simulation and real world.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Notes on contributors

Ryosuke Matsuo received his M.E. from the University of Tokyo, Japan, in 2019. He is currently a Ph.D. candidate at the University of Tokyo. His research interests include machine learning for robot control.

Shinya Yasuda earned his Ph.D. degree from the University of Tokyo, Japan, in 2016. He joined NEC Corporation in 2016, and has been a Senior Researcher at System Platform Research Laboratories since 2018. His research interest includes robotics and networked control.

Taichi Kumagai received the M.E. degree from the University of Electro-Communications, Japan, in 2008. He joined NEC Corporation in 2009, and has been a Senior Researcher at System Platform Research Laboratories since 2017. His research interest includes wireless communication and robotics.

Natsuhiko Sato received his M.S. degree from University of Tokyo, Japan, in 2015. He joined NEC Corporation in 2019, and has been an Associate Researcher at System Platform Research Laboratories. His research interest includes robotics and networked control.

Hiroshi Yoshida received the B.E. and M.E. degrees from Osaka University, Japan, in 2002 and 2004, respectively, and Ph.D. degree from Tokyo Institute of Technology, Japan, in 2016. He joined NEC Corporation in 2004, and has been a Senior Principal Researcher of System Platform Research Laboratories since 2020. His current research interests include robotics and network control.

Takehisa Yairi is currently a professor at the Research Center for Advanced Science and Technology (RCAST) of the University of Tokyo. He received his B.Eng., M.Sc., and Ph.D. degrees in aerospace engineering from the University of Tokyo, Japan in 1994, 1996, and 1999 respectively. His research interests include anomaly detection, health monitoring, fault diagnosis, learning dynamical systems, nonlinear dimensionality reduction, as well as applications of machine learning and probabilistic inference to aerospace systems.

References

[1] Kumagai T, Yasuda S, Yoshida H. A prototype of a cooperative conveyance system by wireless-network control of multiple robots. IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society; Vol. 1, IEEE; 2019. p. 231–236.

[2] Lillicrap TP, Hunt JJ, Pritzel A, et al. Continuous control with deep reinforcement learning. Preprint 2015. Available from: arXiv:1509.02971.

[3] Schulman J, Levine S, Abbeel P, et al. Trust region policy optimization. International conference on machine learning; PMLR; 2015. p. 1889–1897.

[4] Mnih V, Puigdomenech Badia A, Mirza M, et al. Asynchronous methods for deep reinforcement learning. International conference on machine learning; PMLR; 2016. p. 1928–1937.

[5] Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms. Preprint 2017. Available from: arXiv:1707.06347.
[6] Fujimoto S, Hoof H, Meger D. Addressing function approximation error in actor-critic methods. International Conference on Machine Learning; PMLR; 2018. p. 1587–1596.
[7] Haarnoja T, Zhou A, Abbeel P, et al. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. International Conference on Machine Learning; PMLR; 2018. p. 1861–1870.
[8] Todorov E, Erez T, Tassa Y. Mujoco: a physics engine for model-based control. 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems; IEEE; 2012. p. 5026–5033.
[9] Coumans E, Bai Y. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016–2021. Available from: http://pybullet.org.
[10] Rohmer E, Singh SPN, Freese M. V-rep: a versatile and scalable robot simulation framework. 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems; IEEE; 2013. p. 1321–1326.
[11] Koenig N, Howard A. Design and use paradigms for gazebo, an open-source multi-robot simulator. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566); Vol. 3, IEEE; 2004. p. 2149–2154.
[12] Juliani A, Berges V-P, Teng E, et al. Unity: a general platform for intelligent agents. preprint 2018. Available from: arXiv:1809.02627.
[13] Nair A, Srinivasan P, Blackwell S, et al. Massively parallel methods for deep reinforcement learning. International Conference on Machine Learning Deep Learning Workshop; 2015.
[14] Horgan D, Quan J, Budden D, et al. Distributed prioritized experience replay. International Conference on Learning Representations (ICLR); 2018.
[15] Espeholt L, Soyer H, Munos R, et al. Impala: scalable distributed deep-rl with importance weighted actor-learner architectures. International Conference on Machine Learning; PMLR; 2018. p. 1407–1416.
[16] Johannink T, Bahl S, Nair A, et al. Residual reinforcement learning for robot control. 2019 International Conference on Robotics and Automation (ICRA); IEEE; 2019. p. 6023–6029.
[17] Silver T, Allen K, Tenenbaum J, et al. Residual policy learning. Preprint 2018. Available from: arXiv:1812.06298.
[18] Zeng A, Song S, Lee J, et al. Tossingbot: learning to throw arbitrary objects with residual physics. IEEE Trans Robot. 2020;36(4):1307–1319.
[19] Li T, Geyer H, Atkeson CG, et al. Using deep reinforcement learning to learn high-level policies on the atrias biped. 2019 International Conference on Robotics and Automation (ICRA); IEEE; 2019. p. 263–269..
[20] Nguyen TT, Nguyen ND, Nahavandi S. Deep reinforcement learning for multiagent systems: a review of challenges, solutions, and applications. IEEE Trans Cybern. 2020;50(9):3826–3839.
[21] Lowe R, Wu Y, Tamar A, et al. Multi-agent actor-critic for mixed cooperative-competitive environments. Preprint 2017. Available from: arXiv:1706.02275.
[22] Gupta JK, Egorov M, Kochenderfer M. Cooperative multi-agent control using deep reinforcement learning. International Conference on Autonomous Agents and Multiagent Systems; Springer; 2017. p. 66–83.
[23] Tuci E, Alkilabi MHM, Akanyeti O. Cooperative object transport in multi-robot systems: a review of the state-of-the-art. Frontiers in Robotics and AI. 2018;5:185.
[24] Ohashi F, Kaminishi K, Figueroa J, et al. Transportation of a large object by small mobile robots with handcarts and outrigger. 2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014); IEEE; 2014. p. 70–75.
[25] Kosuge K, Oosumi T. Decentralized control of multiple robots handling an object. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS'96; Vol. 1, IEEE; 1996. p. 318–323.
[26] Stroupe A, Huntsberger T, Okon A, et al. Behavior-based multi-robot collaboration for autonomous construction tasks. 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems; IEEE; 2005. p. 1495–1500.
[27] Machado T, Malheiro T, Monteiro S, et al. Multi-constrained joint transportation tasks by teams of autonomous mobile robots using a dynamical systems approach. 2016 IEEE international conference on robotics and automation (ICRA); IEEE; 2016. p. 3111–3117.
[28] Yufka A, Ozkan M. Formation-based control scheme for cooperative transportation by multiple mobile robots. Int J Adv Robotic Syst. 2015;12(9):120.
[29] Brown RG, Jennings JS. A pusher/steerer model for strongly cooperative mobile robot manipulation. Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems; Human Robot Interaction and Cooperative Robots. Vol. 3, IEEE; 1995. p. 562–568.
[30] Wang Z, Hirata Y, Kosuge K. Control multiple mobile robots for object caging and manipulation. Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453); Vol. 2, IEEE; 2003. p. 1751–1756.
[31] Wan W, Shi B, Wang Z, et al. Multirobot object transport via robust caging. IEEE Transactions on Systems, Man, and Cybernetics: Systems. 2017;50(1):270–280.
[32] Wan W, Sun C, Yuan J. The caging configuration design and optimization for planar moving objects using multi-fingered mechanism. Adv Robot. 2019;33(18):925–943.
[33] Rimon E, Blake A. Caging planar bodies by one-parameter two-fingered gripping systems. Int J Rob Res. 1999;18(3):299–318.
[34] Pipattanasomporn P, Sudsang A. Two-finger caging of concave polygon. Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006; IEEE; 2006. p. 2137–2142.
[35] Allen TF, Burdick JW, Rimon E. Two-finger caging of polygonal objects using contact space search. IEEE Trans Robot. 2015;31(5):1164–1179.
[36] Makita S, Wan W. A survey of robotic caging and its applications. Adv Robot. 2017;31(19–20):1071–1085.
[37] Zhang L, Sun Y, Barth A, et al. Decentralized control of multi-robot system in cooperative object transportation using deep reinforcement learning. IEEE Access. 2020;8:184109–184119.

[38] Woo J, Yu C, Kim N. Deep reinforcement learning-based controller for path following of an unmanned surface vehicle. Ocean Engineering. 2019;183:155–166.
[39] Cai P, Mei X, Tai L, et al. High-speed autonomous drifting with deep reinforcement learning. IEEE Robot Autom Lett. 2020;5(2):1247–1254.
[40] Kanayama Y, Kimura Y, Miyazaki F, et al. A stable tracking control method for an autonomous mobile robot. Proceedings., IEEE International Conference on Robotics and Automation; IEEE; 1990. p. 384–389.
[41] Bergstra J, Bardenet R, Bengio Y, et al. Algorithms for hyper-parameter optimization. 25th annual conference on neural information processing systems (NIPS 2011); Vol. 24, Neural Information Processing Systems Foundation; 2011.
[42] Akiba T, Sano S, Yanase T, et al. Optuna: a next-generation hyperparameter optimization framework. Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining; 2019. p. 2623–2631.
[43] Silver D, Lever G, Heess N, et al. Deterministic policy gradient algorithms. International conference on machine learning; PMLR; 2014. p. 387–395.
[44] Schaul T, Quan J, Antonoglou I, et al. Prioritized experience replay. International Conference on Learning Representations (ICLR); 2016.
[45] Kingma DP, Ba J. Adam: a method for stochastic optimization. ICLR (Poster). 2015.

Appendix A. Details of formation-based feedback controller formulation

The calculation process of the formation-based feedback controller is written below. The positions defined in the below equation are in local coordinates at time k.

$$\begin{aligned}\theta^{\mathrm{sr}}_{\mathrm{tgt}}(k) &= \pi + \tan^{-1}\frac{y^{\mathrm{sr}}_{\mathrm{tgt}}(k) - y^{\mathrm{sr}}_{\mathrm{tgt}}(k-1)}{x^{\mathrm{sr}}_{\mathrm{tgt}}(k) - x^{\mathrm{sr}}_{\mathrm{tgt}}(k-1)} \\ &= \pi + \tan^{-1}\frac{y^{\mathrm{sr}}_{\mathrm{tgt}}(k) - y^{\mathrm{sr}}_{\mathrm{tgt}}(k-1)}{x^{\mathrm{sr}}_{\mathrm{tgt}}(k) - x^{\mathrm{sr}}_{\mathrm{tgt}}(k-1)} \\ &\quad - \tan^{-1}\frac{y^{\mathrm{VL}}_{\mathrm{tgt}}(k-1)}{x^{\mathrm{VL}}_{\mathrm{tgt}}(k-1)} + \tan^{-1}\frac{y^{\mathrm{VL}}_{\mathrm{tgt}}(k-1)}{x^{\mathrm{VL}}_{\mathrm{tgt}}(k-1)} \\ &= \pi + \tan^{-1}\frac{-R(1-\cos\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}}) + L\sin\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}}}{R\sin\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}} + L(1-\cos\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}})} \\ &\quad + \tan^{-1}\frac{R\left(1-\cos\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}}\right)}{R\sin\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}}} - \frac{\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}}}{2} \\ &= \pi + \tan^{-1}\left(-\frac{(1-\cos\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}}) + \rho L\sin\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}}}{\sin\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}} + \rho L(1-\cos\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}})}\right) \\ &\quad + \tan^{-1}\frac{1-\cos\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}}}{\sin\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}}} - \frac{\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}}}{2} \\ &= \pi + \rho L - \frac{\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}}}{2} \end{aligned} \tag{A1}$$

The calculation from lines 4 to 5 utilizes the sum of angle identities. L is defined as in Figure 3 and the positions are defined as

$$\begin{aligned}\boldsymbol{x}^{\mathrm{sr}}_{\mathrm{tgt}}(k) &= \left[x^{\mathrm{sr}}_{\mathrm{tgt}}(k), y^{\mathrm{sr}}_{\mathrm{tgt}}(k)\right] = [L, 0]^{\top}, \\ \boldsymbol{x}^{\mathrm{sr}}_{\mathrm{tgt}}(k-1) &= \boldsymbol{x}^{\mathrm{VL}}_{\mathrm{tgt}}(k-1) + \left[L\cos\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}}, -L\sin\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}}\right]^{\top} \\ &= \Big[-R\sin\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}} + L\cos\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}}, \\ &\qquad R(1-\cos\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}}) - L\sin\omega^{\mathrm{VL}}_{\mathrm{tgt}}t_{\mathrm{s}}\Big]^{\top}.\end{aligned}$$