Multi-robot Benchmark for Collaborative Manipulation Tasks

Seonghyun Kim
Digital Convergence Research Laboratory
Electronics and Telecommunications Research Institute
Daejeon, Korea
kim-sh@etri.re.kr

Ingook Jang
Digital Convergence Research Laboratory
Electronics and Telecommunications Research Institute
Daejeon, Korea
ingook@etri.re.kr

Samyeul Noh
Digital Convergence Research Laboratory
Electronics and Telecommunications Research Institute
Daejeon, Korea
samuel@etri.re.kr

Donghun Lee
Digital Convergence Research Laboratory
Electronics and Telecommunications Research Institute
Daejeon, Korea
donghun@etri.re.kr

Heechul Bae
Digital Convergence Research Laboratory
Electronics and Telecommunications Research Institute
Daejeon, Korea
hessed@etri.re.kr

***Abstract*— Recently, research on multi-agent environments has been continuously conducted based on the development of research on single-agent environments. This paper deals with the study of collaborative task design and learning for multiple robots to cooperate in an environment where multiple robots operate. Unlike a single robot environment, the multi-robot environment has a non-stationary characteristic due to the relationship between robot actions affecting each other. Such non-stationary problems are highly complex and related to collaboration tasks because they affect the performance degradation of learning and the convergence time of learning models. In this paper, we present three types of collaborative task problems: Pick-Push-Place, Collaborative Lift, and Handover. We also discuss the implementation of each task and the validation of learning outcomes.**

Keywords—multi-agent reinforcement learning, multi-robot benchmark, collaborative manipulation task.

I. Introduction

Recently, reinforcement learning has been studied in various environments such as 2D games and 3D object pose control and has made significant progress. Technological advances in reinforcement learning are also expanding as an attempt to introduce reinforcement learning in robot environments. Robot environments correspond to problems with high difficulty in control problems [1]-[7]. While many attempts and algorithms have been proposed to solve manipulation tasks such as Reach, Push, Place in single robot environments, algorithm development that operates stably for collaborative tasks in multi-robot environments has not been developed relatively compared to single robot tasks.

Unlike single-robot environments, multi-robot environments have a non-stationary nature in which the stochastic changes in a multi-robot environment vary from hour to hour because the actions of the robots affect each other [8]. For example, from the perspective of a particular robot, the probability of the transition of the environment state due to the action of that robot changes along the rows of other robots because the action of other robots affects the outcome of that robot's action. This means that more empirical data is required than in a single-robot environment due to the wider range of changes in the environment state space, which can lead to problems such as delayed convergence time of the learning model and poor learning performance. The complexity of these problems is closely related to the design of the working environment. In this paper, we present three types of collaborative task problems: Pick-Push-Place, Collaborative Lift, and Handover. We also discuss the implementation of each task and the validation of learning outcomes.

This work was supported by Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean government. [23ZR1100, A Study of Hyper-Connected Thinking Internet Technology by autonomous connecting, controlling, and evolving ways].

II. Collaborative Manipulation Tasks

In this section, we present three kinds of work related to collaborative manipulation tasks. The robot used in all tasks consists of an arm with 6-Degrees of freedom (DoF) and a gripper with 1-DoF. The detailed configuration and reward functions for each task are described as follows.

A. Pick-Push-Place

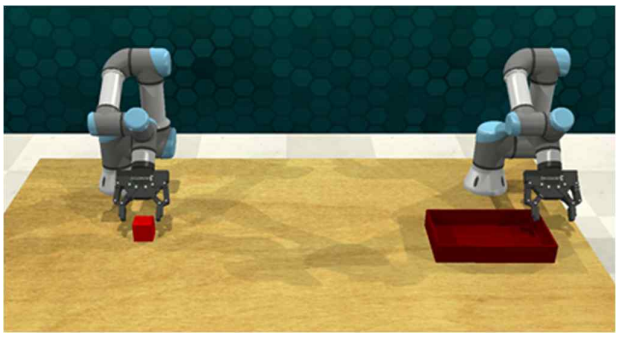


Fig. 1. An environment for the Pick-Push-Place task.

Figure 1 shows the environment for the Pick-Push-Place task, which consists of two robots and two target objects (a red cube and a maroon tray). In the Pick-Push-Place task, the success condition is defined as "the cube is placed inside the tray". To satisfy the success condition in this environment, the robot on the left grasps the cube, the robot on the right grasps the tray, and the two robots collaboratively place the objects by moving their respective grasps to the center of the table.

The reward functions for learning in the Pick-Push-Place task are defined as follows:

- Grasping reward:
$$r^i_{g,o^i} = \begin{cases} c_g, & \text{for robot } i \text{ grasping object } o^i \\ 0, & \text{for otherwise} \end{cases}$$
- Distance reward for robots:
$$r^i_{d,o^i} = -\sqrt{\left\|\mathbf{b}^i_p - \mathbf{o}^i_p\right\|^2}$$
- Distance reward for objects:
$$r_{d,o} = -\sqrt{\left\|\mathbf{o}^1_p - \mathbf{o}^2_p\right\|^2}$$
- Place reward:
$$r_p = \begin{cases} c_p, & \text{for the state when cube is in tray} \\ 0, & \text{for otherwise} \end{cases}$$

where i is the index of the robot, $\mathbf{b}^i_p$ is the robot i's gripper position, $\mathbf{o}^i_p$ is the object position, and c_g and c_p are positive constants. Using these reward functions, a total reward for Pick-Push-Place task at time t is defined as

$$r^{ppp}_t = \sum_i \left(r^i_{g,o^i} + r^i_{d,o^i}\right) + r_{d,o} + r_p \ . \quad (1)$$

B. Collaborative Lift

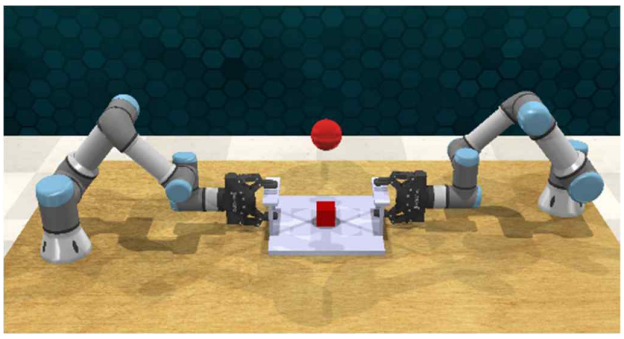


Fig. 2. An environment for a Collaborative Lift task.

Figure 2 shows an environment for a Collaborative Lift task consisting of two robots, a target object (red cube), a goal position (red sphere), and a tray with a handle (gray tray). To succeed in this environment, the two robots must grasp each a handle of the tray, then lift the tray while keeping it horizontal to move the target object to the goal position.

The reward functions for learning in the Collaborative Lift task are defined as follows:

- Grasping reward:
$$r^i_{g,h^i} = \begin{cases} c_g, & \text{for robot } i \text{ grasping handle } h^i \\ 0, & \text{for otherwise} \end{cases}$$
- Distance reward for robots:
$$r^i_{d,h^i} = -\sqrt{\left\|\mathbf{b}^i_p - \mathbf{h}^i_p\right\|^2}$$
- Distance reward for object:
$$r^g_{d,o} = -\sqrt{\left\|\mathbf{g}_p - \mathbf{o}_p\right\|^2}$$
- Lift reward:
$$r_l = \begin{cases} c_l, & \text{for the state when cube reaches goal} \\ 0, & \text{for otherwise} \end{cases}$$

where $\mathbf{h}^i_p$ is the handle position, $\mathbf{g}_p$ is the goal position, and c_l is a positive constant. Using these reward functions, a total reward for Collaborative Lift task at time t is defined as

$$r^d_t = \sum_i \left(r^i_{g,h^i} + r^i_{d,h^i}\right) + r^g_{d,o} + r_l \ . \quad (2)$$

C. Handover

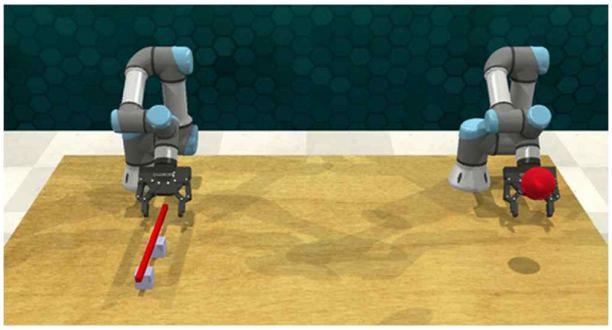


Fig. 3. An environment for the Handover task.

Figure 3 shows the environment for the Handover task, which consists of two robots, a target object (red stick), a goal position (red sphere), and a dummy object (gray cube). In the handover task, the success condition is defined as "the target object reaches the goal position". To satisfy the success condition in this environment, the robot on the left must grasp the target object, pass it to the robot on the right, and then the robot on the right must move the target object to the goal position.

The reward functions for learning in the Handover task are defined as follows:

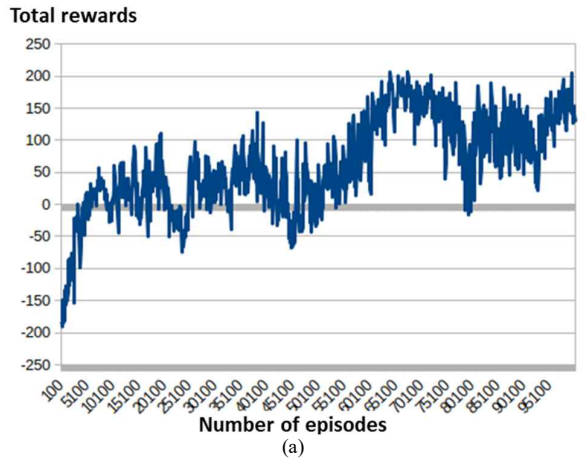
- Grasping reward:
$$r^i_g = \begin{cases} c_g, & \text{for gripper grasping target object} \\ 0, & \text{for otherwise} \end{cases}$$
- Distance reward for robots:
$$r^i_{d,o} = -\sqrt{\left\|\mathbf{b}^i_p - \mathbf{o}_p\right\|^2}$$
- Distance reward for object:
$$r^g_{d,o} = -\sqrt{\left\|\mathbf{g}_p - \mathbf{o}_p\right\|^2}$$
- Handover reward:
$$r_h = \begin{cases} c_h, & \text{for grasping reward pair } (0, c_g) \\ 0, & \text{for otherwise} \end{cases}$$

where c_h is a positive constant. Using these reward functions, a total reward for Handover task at time t is defined as

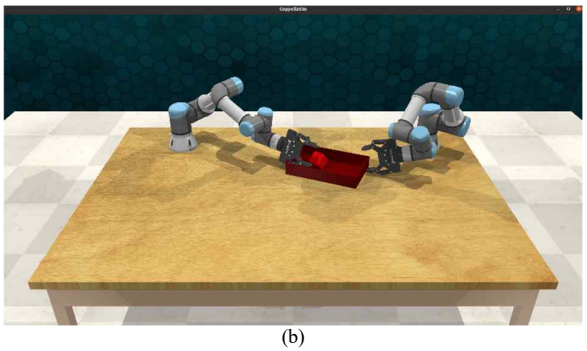
$$r^{ho}_t = \sum_i \left(r^i_g + r^i_{d,o}\right) + r^g_{d,o} + r_h \ . \quad (3)$$

D. Dec-POMDP

To deal with non-stationary problems in multi-agent environments, it is essential to extend Markov decision processes (MDPs) to distributed partially observable Markov decision processes (Dec-POMDPs). In a Dec-POMDP, the distributed partial observation setting means that the probability of a state transition consists of multiple actions and observations, under the assumption that each agent knows only its own actions and not the actions of other agents.

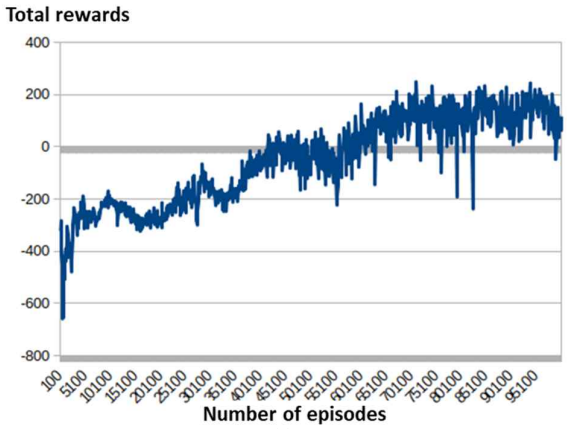


(a)

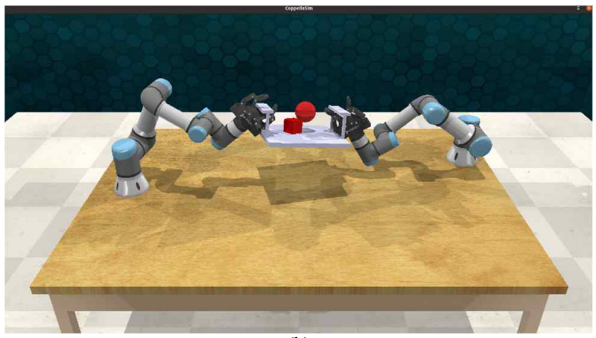


(b)

Fig. 4. Task verifications for the Pick-Push-Place task, (a) total rewards according to episodes, (b) a screenshot at the task termination.



(a)



(b)

Fig. 5. Task verifications for the Collaborative task, (a) total rewards according to episodes, (b) a screenshot at the task termination.

E. Multi-Agent Deep Deterministic Policy Gradient

In this work, we adopt Multi-Agent Deep Deterministic Policy Gradient (MA-DDPG) as the baseline for the validation of three kinds of benchmark tasks. MA-DDPG uses a centralized method in the training phase and a distributed method in the execution phase to solve non-stationary problems in multi-agent environments [10]. MA-DDPG is based on the Actor-Critic method, where the policies of multiple agents corresponding to actors are executed and updated in a distributed manner, and the value functions corresponding to critics are updated in a centralized manner. The value function is updated through a centralized method given all the action information of the agents, so the non-stationary problem does not affect it. Since each agent's policy is learned through this centralized value function and does not take into account the actions of other agents, multiple agents act in a distributed manner.

III. Simulation Result

A. Environment Implemenation

The collaborative task environments are implemented based on the Robot Learning Benchmark and Learning Environment (RLBench) [11]. RLBench provides an interface to easily create different task models. RLBench was developed based on Copeliasim and Pyreb. Coppeliasim was developed to facilitate the design and verification of robotic algorithms, robot simulation and training, and virtual world-based safety checks. PyRep is a toolkit for robot learning that utilizes CoppeliaSim and was developed to run CoppeliaSim in Python.

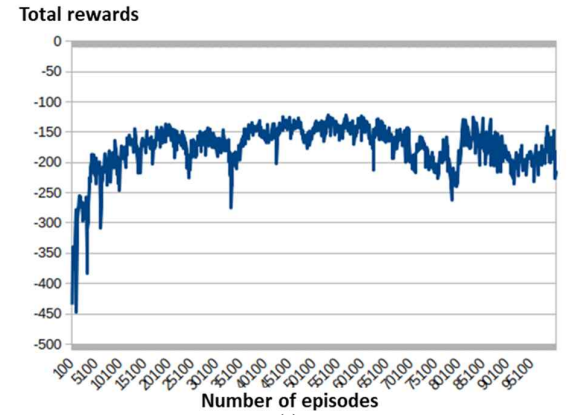
RLBench provides graphical user interface-based design tools and functionality for users who want to design new robot manipulation tasks. Designing a task requires two details: an environment model and a description. The environment model contains all three-dimensional scene information, such as the positions and properties of objects. The description includes the success condition for the task, and the definition of rewards function for the robot actions.

B. Network Implemenation and Learning Setting

The implementation of MA-DDPG is identical to the structure and hyperparameters in [10]. Each policy and value function is parameterized by a two-layer ReLU MLP with 64 units per layer. During the learning process, the maximum length per episode is set to 50 and the maximum number of episodes to 50000.

C. Task Verification

Figure 4 shows task verifications for the Pick-Push-Place task, where Figures 4(a) and (b) represent total rewards according to episodes and a screenshot at the task termination, respectively. As shown in Figure 4(a), performance increases steeply with increasing the distance reward until about 10000 episodes, and then goes through a phase of learning the grasping reward until about 50000 episodes. At about 65000 episodes, the maximum performance is achieved with the place reward, after which the performance drops due to further exploration and then increases again. As shown in Figure4(b), the Pick-Push-Place task is completed by collaboration between the two robots.



(a)

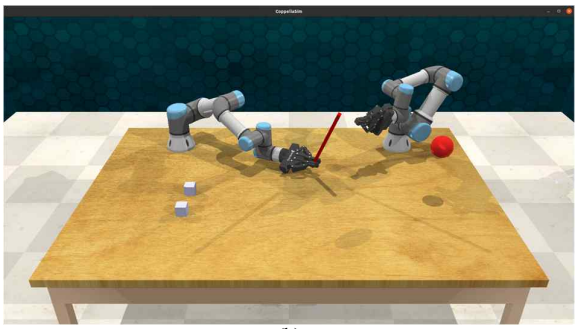

(b)

Fig. 6. Task verifications for the Handover task, (a) total rewards according to episodes, (b) a screenshot at the task termination.

Figure 5 shows the task validation for Collaborative Lift task. As shown in Figure 5(a), the performance increases rapidly as the grasping reward increases until about 10000 episodes, then it goes through a phase where it tries to learn the lift reward until about 35000 episodes. After about 35000 trials, it can be seen that the lift reward gradually increases as the two robots collaborate to balance the tray. At around 70000 iterations, the maximum performance is reached and convergence is shown. As shown in Fig. 5(b), the Collaborative Lift Task is completed by the collaboration of the two robots to balance the tray.

Figure 6 shows the task validation for the handover task. As observed in Figure 6(a), the performance initially increases but reaches a limit. More specifically, the first robot is able to grasp the target object, but the second robot is not able to grasp the target object, i.e., no handover is performed. One of the reasons why the handover task is difficult to solve with learning is the credit assignment problem. Since handovers are performed by acting in different directions on the same object, it is difficult to determine how much each robot's action contributed to the reward in terms of a single summed reward, i.e., the credit assignment problem. Another reason is that we need to look at it from the perspective of the first robot. Handover means that the first robot gives up the grasping reward on its own. As distributed actors, the robots learn their actions based on optimization in a greedy manner, so during the learning process, the first robot does not give up the grasping reward and the handover is not completed.

IV. Conclusion

In this paper, we define three types of collaborative tasks, including Pick-Push-Place, Collaborative Lift, and Handover, and verify them using the base algorithm MA-DDPG. The simulation results show that the simple application of MA-DDPG is limited in solving collaborative tasks that are highly affected by the credit assignment problem, such as handover. In future work, we would like to conduct research to solve credit assignment problems in collaborative tasks.

References

[1] Y.-J. Han I.-S. Kim and Y.-D. Hong, "Optimization-based humanoid robot navigation using monocular camera within indoor environment," ETRI Journal, vol. 40, no. 4, pp. 446-457, Aug., 2018.

[2] M. Zhang, J. Chen, X. Wei, and D. Zhang, "Work chain-based inverse kinematics of robot to imitate human motion with Kinect," ETRI Journal, vol. 40, no. 4, pp. 511-521, Aug., 2018.

[3] M. Ilbeygi *et al*, "Comprehensive architecture for intelligent adaptive interface in the field of single-human multiple-robot interaction," ETRI Journal, vol. 40, no. 4, pp. 483–498, Aug., 2018.

Canovas, Bruce, Amaury Negre, and Mich ` ele Rombaut. "Onboard dynamic RGB-D simultaneous localization and mapping for mobile robot navigation." ETRI Journal 43.4 (2021): 617-629.

[4] S. Jung, et al. "Collision-free local planner for unknown subterranean navigation," ETRI Journal pp. 580-593, 43(4), 2021.

[5] S. Hong. et al. "Special issue on recent advancements in simultaneous localization and mapping (SLAM) and its applications." ETRI Journal pp. 577-579, 43(4), 2021.

[6] S. Seo, and H. Jung. "A robust collision prediction and detection method based on neural network for autonomous delivery robots." ETRI Journal pp. 329-337, 45(2), 2023.

[7] W. Yu, and S. Song, "Design and experimentation of remote driving system for robotic speed sprayer operating in orchard environment," ETRI Journal pp. 479-491, 45(3), 202),.

[8] H. Kim *et al*, "Avoiding collaborative paradox in multi-agent reinforcement learning," ETRI Journal pp. 1004-1012, 43(6), 2021.

[9] R. Lowe *et al*, "Multiagent actor-critic for mixed cooperative-competitive environments," in Proc. NIPS, pp. 2094–2100, 2017.

[10] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark learning environment," IEEE Robotics and Automation Letters, vol. 5, no. 2, Apr., 2020.