MRDRL-ROS: a Multi Robot Deep Reinforcement Learning Platform based on Robot Operating System

Zhongxuan Cai, Zhen Liang and Jing Ren*

Institute for Quantum Information & State Key Laboratory of High Performance Computing, College of Computer, National University of Defense Technology, Changsha, Hunan, 410073, China

*Corresponding author’s e-mail: renjing@nudt.edu.cn

Abstract. Deep reinforcement learning (DRL) has greatly improved the intelligence of AI in recent years and the community has proposed several common software to facilitate the development of DRL. However, in robotics the utility of common DRL software is limited and the development is time-consuming due to the complexity of various robot software. In this paper, we propose a software engineering approach leveraging modularity to facilitate robot DRL development. The platform decouples learning environment into task, simulator and hierarchical robot modules, which in turn enables diverse environment generation using existing modules as building blocks, regardless of the underlying robot software details. Experimental results show that our platform provides composable environment building, introduces high module reuse and efficiently facilitates robot DRL.

1. Introduction

Deep reinforcement learning (DRL) has been one of the most important breakthroughs of artificial intelligence in recent years. The first hallmark was the Deep Q-Network (DQN) achieving superhuman performance on Atari 2600 video games in 2013 [1]. Apart from the field of video games, DRL is also widely applied in robotics, where robot control tasks can be naturally modeled as sequential decision problems, i.e. to determine an action sequence based on sensor states. The feature extraction ability of deep neural networks enables end-to-end control from robot sensors to actuators, developing multi-joint motion control [2], mobile robot navigation [3], 3D manipulation [4], etc. DRL common software is developed in the community, providing benchmark environments [5], DRL platforms [6] and parallel learning architectures [7].

Although the software infrastructure for general reinforcement learning is increasingly sophisticated, robot reinforcement learning faces additional problems arising from the unique characteristics of the robotics field, i.e. the software complexity of environment building. Distinct from general DRL development, robot DRL aims to accomplish a variety of robot tasks with various robot hardware rather than solving a limited set of benchmarks. The environment building requires customization. It usually involves third party robot simulators and communicates with multitudinous robot sensors and actuators like GPS, laser, sonar, camera, mechanical arms, mobile bases, rotors and so on [8], which makes the program extremely complex and produces a huge workload, especially for multi-robot cases. The software complexity makes it rather inefficient and time consuming to develop robot DRL.

In this paper, we propose a robot DRL platform design as the approach to robot DRL software engineering. Following modularity, we abstract robot DRL environment into three reusable modules...
without loss of generality: the simulation, robot and task modules. The simulation module adapts to different robot simulators and provides easy transfer through module switching. The robot module encapsulates communication details with various robot sensors and actuators, by which custom robot groups can be built without re-engineering underlying details. The task module implements a specific robot task. By modularity, custom learning environments with arbitrary combination of simulators, robot groups and tasks can be built like building blocks.

The major contributions of this paper are:
- We propose a software engineering approach to building robot deep reinforcement learning programs, which addresses the software complexity of robot DRL through modularity from software infrastructure.
- The modularity of proposed model abstracts the environment into three high-level reusable modules and decouples the engineering of robot simulators, robot groups and tasks, which supports customizing environment like building blocks, regardless of underlying details.
- Experiments with different robot type, different simulator and different tasks are conducted to verify the effectiveness of the proposed platform.

2. Related Work
With the development of deep reinforcement learning, several software platforms are proposed to improve the efficiency of DRL development and free researchers from tedious works unrelated to DRL algorithm design, e.g. the Arcade learning environment provided an interface to hundreds of Atari 2600 games as learning environment [5]. Besides out-of-box benchmark environments, the community further considered efficient programming in DRL software engineering. OpenAI gym [6] proposed unified interacting interfaces between DRL algorithms and environments, allowing cross-testing of different algorithms and environments as long as they followed the gym interface specification. Gym's modular design has built a powerful algorithm and environment software ecosystem, making it the de facto standard for DRL research and development.

While the common software for general DRL has developed a lot, its usage and utility in robotics is limited. In robot DRL, the ultimate goal is to achieve multitudinous robot tasks given certain robot hardware, rather than to merely achieve higher performance on fixed benchmark. The environment building needs to be customized and third party robot simulators are usually involved for physics simulation. Gym-gazebo and parallel gym gazebo [7] extend openAI gym with gazebo, a powerful 3D robot simulator. However, they extend fixed environments instead of providing a mechanism to efficiently build diverse robot tasks.

3. The MRDRL Platform
To improve the efficiency of building environments for multitudinous robot DRL tasks, we propose a modular environment model which can be built from abstraction with the above three parts. The modularity provides high code reuse, convenient transfer and ease of environment modification.

3.1. Model architecture
The architecture decouples the environment model for robot DRL into three parts, including simulation, robots and task modules, as shown in Fig. 1.

The environment model bridges communication between DRL algorithms and robot simulators or hardware drivers. On the top of the architecture, the environment interfaces of openAI gym (reset and step) are preserved between the DRL algorithm and the environment to be compatible with gym-based DRL algorithm implementations. At the bottom of the architecture, the environment model integrates rich robot simulators or hardware drivers through ROS communication.

The main idea of our design is the building blocks philosophy. With a set of implemented simulation, robots and task modules, an environment can be efficiently constructed or modified by combing these modules. A set of robots can perform a different learning task, e.g. from exploring a
maze to formation, by simply modifying the task module rather than rewriting a completely new environment.

![Figure 1. The Robot DRL platform architecture.](image)

### 3.2. Modularized learning procedure

In this subsection we describe the function of each module in our environment architecture through a top-down approach. Since the architecture keeps the unified environment interfaces of openAI gym, from the perspective of DRL, all the environment model needs to accomplish is to reset and step the simulation world and return feedback through ROS communication.

Rather than directly implementing the reset and step details in a flat process-oriented way for each robot learning task, our model resets and steps the environment through three abstract modules as shown in Fig. 2. To reset the environment, the simulation module is called to reset and unpause the underlying robot simulator. Then the robots module is utilized to wait for sensor messages of each robot from ROS as the raw observation. Once these data received, the simulation is paused by the simulation module, and the raw sensor data is managed by the task module to produce observation in float lists directly available to DRL neural networks.

![Figure 2. Modularization of environment interaction interface.](image)

Similarly, the step procedure is also achieved with the three proposed modules in Fig. 2. The additional parts includes: (1) sending action command messages by the robots module, which converts the action calculated by DRL algorithms to ROS communication messages and publishes them to
corresponding robot actuators, and (2) the task module calculating the reward and the episode-end flag as the step feedback. Given the required functions of each module to accomplish environment reset and step, we design the modules in the following subsections.

3.3. Simulation module for transfer
During environment reset and step, the major function of the simulation module is to reset, pause and unpause the underlying robot simulator. The reset function is utilized to restart a new simulation world for a new episode. The pause and unpause function is needed to accomplish frame simulation and learning step by step. The simulation module encapsulates all the communication details of simulator control. When building environment for new robot tasks with the same simulator, the implemented module can be directly reused and no underlying detail of simulator control needs to be taken care of. After several mainstream simulator implemented, transfer between these simulators can be enabled by simply changing to corresponding modules, where all communication details are handled by these modules and hidden from developers.

3.4. Hierarchical robots module for communications
The robots module is the core of communication between DRL algorithms and robots. It inputs DRL results as action commands and sends action command messages to each robot actuator, then waits for sensor messages from all robots and returns observations. The major function of the robots module is the conversion of raw action/observation data and messaging with actuators and sensors.

It is natural to organize the group from the group level to robot level to actuator/sensor level as in Fig. 1. The operation of the robots module can be achieved by iteratively querying each robot as submodules to operate. Each robot submodule also iteratively queries its actuators and sensors to accomplish the query of the robot. With the query structure, the upper robot modules can be generated by listing child modules like actuators and sensors. In ROS different robot sensors and actuators communicate on separate topics but the message format is the same for each type. We develop a namespace mechanism by which the communication detail of a specific type of sensor or actuator can be reused and redirected to different channels. As shown by the same colored submodules in Fig. 3, when managing different actuators or sensors of the same type, we can assign namespace to communicate on different channels and reuse existing implementations. The higher robot module of the same type can also be reused through the namespace mechanism.

After implementing all needed types of actuators and sensors once, we can build robot groups by simply combing these modules and assigning namespaces, regardless of any communication details.
The complexity of the underlying communication management only increases with the types of sensors and actuators

3.5. Task module for learning
Distinct from the simulation module and the robots module which handle communication with the simulators and robot drivers, the task module encapsulates the reward signal for robots to accomplish a specific learning task. It processes raw sensor data to task-defined environment observations that match the input of DRL neural network. Besides, the task module also determines the terminate condition for an episode and returns the episode-end flag for further environment reset.

With the proposed reusable task, simulation and robots modules, environment building of robot deep reinforcement learning is decoupled and modularized. After implementing these modules for once, developers are free to build arbitrary learning environment of any task on any group of robots in any simulator by simply combining these high-level modules like building blocks.

4. Experiments

To validate our work, we conduct several robot DRL experiments with a widely used DRL algorithm, proximal policy optimization (PPO), which performs well on both continuous and discrete problems [9]. We build modular environment models and demonstrate modifying simulators, robots and task modules to efficiently switch to new learning environment. The basic environment is the standard benchmark of gym-gazebo, which contains a turtlebot 2 mobile robot which learns to explore in a maze without colliding walls, as shown in Fig. 4.

To build a modular environment model between the DRL algorithm and the gazebo robot simulator, a gazebo simulation module contains communication details to pause, unpause and reset the gazebo simulator. Then we build a hierarchical robots module containing one single turtlebot-laser model which is made up of one actuator model and two sensor models. The actuator model implements communication details with the robot base driver. The laser sensor model and position sensor model implements communication details with the sensor drivers. The task module calculates exploration reward [7]. Since the current learning task only requires laser data as observation, the task module removes the position sensor data from the original sensor data as observation.

To conduct the same learning task through a different simulator, switching to a new simulation module is all that needs in our modular environment model. We demonstrate the modification in Fig. 4(a), where a turtlebot mobile robot learns to explore a maze in Stage robot simulator rather than the gazebo simulator. To build an environment model between the simulator and the DRL algorithm, we load a new simulation module, keeping the existing robots module and task module invariant. The new simulation module for Stage implements communication details for controlling Stage simulator.
Besides modifying simulator, the modularity also enables switching to a new robot group to perform the robot task by modifying the robots module. To build a new robots module, we can further reuse existing robot models, or build new robot models with existing actuator and sensor models. We change the robot group from one single turtlebot to two turtlebots as shown in Fig. 4(b) by simply adding more turtlebot models and assigning distinct namespace. We further add a multi-rotor UAV model consisting of a flight control actuator, laser and position sensors. The simulator we use is the gazebo simulator and the task is still exploration, therefore the two corresponding modules are reused.

Finally, we build a new environment with the same robot simulator and robot group to conduct a new learning task. As shown by Fig. 4(c), we still use two turtelbots and one multi-rotor UAV in the gazebo robot simulator therefore we reuse the existing robots and simulation modules. But rather than exploration, we create a new formation task where each robot aims to navigate and form a triangle formation. The new task module converts position data to relative polar position and concatenate the laser as observation. It also calculates rewards based on Euclidean distance to the target position [10].

5. Conclusion
In this paper, we propose a common software approach for engineering robot DRL programs. The modularity platform enables reuse of modules to avoid inefficient re-engineering of environments and provides high level abstraction to free algorithm designers from underlying robot software details. Experiments are conducted to verify the effectiveness the proposed approach. Future work is to consider the learning procedure improvement of multi-robot cooperative learning.

Acknowledgments
This work was partially supported by the National Key Research and Development Program of China (No. 2017YF-B1301104) and the National Natural Science Foundation of China (No. 91648204 and 61803375).

References
[1] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M.A., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature 518(7540), 529–533 (2015).
[2] Schulman, J., Levine, S., Abbeel, P., Jordan, M.I., Moritz, P.: Trust region policy optimization. In: Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015, vol. 37, pp. 1889–1897.
[3] Hsu, S., Chan, S., Wu, P., Xiao, K., Fu, L.: Distributed deep reinforcement learning based indoor visual navigation. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2018, Madrid, Spain, October 1-5, 2018, pp. 2532–2537.
[4] Gu, S., Holly, E., Lillicrap, T.P., Levine, S.: Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In: 2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, May 29 - June 3, 2017, pp. 3389–3396.
[5] Bellemare, M.G., Naddaf, Y., Veness, J., Bowling, M.: The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research 47, 253–279 (2013).
[6] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: Openai gym. ArXiv abs/1606.01540 (2016).
[7] Liang, Z., Cai, Z., Li, M., Yang, W.: Parallel gym gazebo: a scalable parallel robot deep reinforcement learning platform. In: 31st IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2019, Portland, OR, USA, November 4-6, 2019, pp. 206–213.
[8] Kober, J., Bagnell, J.A., Peters, J.: Reinforcement learning in robotics: A survey. The International Journal of Robotics Research 32(11), 1238–1274 (2013).
[9] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. ArXiv abs/1707.06347 (2017).

[10] Long, P., Fan, T., Liao, X., Liu, W., Zhang, H., Pan, J.: Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In: 2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May