MIRSim-RL: A Simulated Mobile Industry Robot Platform and Benchmarks for Reinforcement Learning

Qingkai Li[1,*], Zijian Ma[2,*], Chenxing Li[3,6,*], Yinlong Liu[4], Tobias Recker[5], Daniel Brauchle[6], Jan Seyler[6], Mingguo Zhao[1] and Shahram Eivazi[3,6]

[1]*Department of Automation, Tsinghua University, Beijing, China*

[2]*Department of Mechanical Engineering, Technical University of Munich, Munich, Germany*

[3]*Department of Computer Science, University of Tübingen, Tübingen, Germany*

[4]*Department of Computer Science, University of Macau, Macau, China*

[5]*Institute of Assembly Technology and Robotics, Leibniz University Hanover, Germany*

[6]*Advanced Develop. Analytics and Control, Festo SE & Co. KG, Esslingen, Germany*

Keywords: Simulated Mobile Robot Platform, Reinforcement Learning, Whole-Body Control.

Abstract: The field of mobile robotics has undergone a transformation in recent years due to advances in manipulation arms. One notable development is the integration of a 7-degree robotic arm into mobile platforms, which has greatly enhanced their ability to autonomously navigate while simultaneously executing complex manipulation tasks. As such, the key success of these systems heavily relies on continuous path planning and precise control of arm movements. In this paper, we evaluate a whole-body control framework that tackles the dynamic instabilities associated with the floating base of mobile platforms in a simulation closely modeling real-world configurations and parameters. Moreover, we employ reinforcement learning to enhance the controller's performance. We provide results from a detailed ablation study that shows the overall performance of various RL algorithms when optimized for task-specific behaviors over time. Our experimental results demonstrate the feasibility of achieving real-time control of the mobile robotic platform through this hybrid control framework.

1 INTRODUCTION

In recent decades, we have seen the widespread adoption of integrating a robotic arm into mobile platforms (Ramalepa and Jr., 2021; Guo et al., 2016). For instance, Uehara et al. (Uehara et al., 2010) proposed a mobile robot with an arm to assist individuals with severe disabilities. Similarly, Grabowski et al. (Grabowski et al., 2021) demonstrated the practical application of such platforms in industrial settings, where they assist workers in performing various tasks.

In mobile platforms equipped with a robotic arm, a critical challenge lies in solving the problem of kinematic trajectory planning for both the arm and the mobile base. A common approach has been to control the arm and the mobile platform independently (Tinós et al., 2006). However, this independent control paradigm is insufficient for platforms with integrated arms, where the interaction between the two systems introduces significant complexity when the platform system is not heavy enough, for instance, when the robot arm moves significantly, the center of gravity of the entire robot platform shifts. This distinction forms the basis of the work presented in this paper. Unlike traditional fixed-base robotic arms, mobile platforms experience dynamic changes in their floating base, which can adversely affect the stability of the overall system. Therefore, a comprehensive controller that simultaneously manages both the arm and the mobile platform is essential to address these stability issues effectively.

A prominent approach to addressing this challenge is the Whole-Body Control (WBC) technique (Dietrich et al., 2012), which unifies the control of both the robotic arm and the mobile platform into a single solution. WBC enables direct control of all joints through joint state commands, allowing for coordinated motion across the entire robotic system. Additionally, alternative methods such as neural net-

*The authors with * contribute equally

Li, Q., Ma, Z., Li, C., Liu, Y., Recker, T., Brauchle, D., Seyler, J., Zhao, M. and Eivazi, S.
MIRSim-RL: A Simulated Mobile Industry Robot Platform and Benchmarks for Reinforcement Learning.
DOI: 10.5220/0013159400003890

In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025) - Volume 1*, pages 56-67
ISBN: 978-989-758-737-5; ISSN: 2184-433X

works (Guo et al., 2021) and reinforcement learning (RL) (Xin et al., 2017) have emerged as powerful tools to simplify the complexity of control strategies. These machine-learning techniques excel at learning and generalizing behaviors in an end-to-end manner, further enhancing the capabilities of robotic control systems.

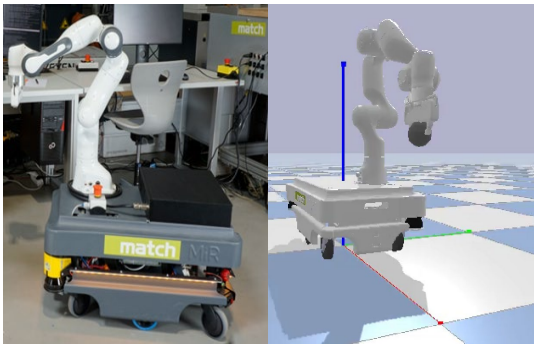


Figure 1: Mobile Industrial Robot and its simulation on PyBullet Engine. The left shows the real mobile robot platform while the right is the simulation platform we built upon Pybullet. (This platform is developed by Institute of Assembly Technology and Robotics, Leibniz University Hanover, Germany).

In this study, we present a novel framework that integrates a mobile robot with an arm (Mobile Industrial Robot[1]) into the PyBullet simulation environment (Coumans and Bai, 2021). We develop a whole-body controller to manage path planning and platform movement, which is designed based on real-world parameters to facilitate efficient sim-to-real transfer. Additionally, we utilize a series of RL algorithms (online, offline, single-agent, and multi-agents) to enhance the controller's performance by learning task-specific behaviors, leading to a more balanced control mechanism. The primary contributions of our work are as follows:

- We develop a simulated mobile robot equipped with a robotic arm, employing a WBC mechanism within the Pybullet environment, integrated through the Gym API (Brockman et al., 2016). This framework enables the testing and validation of algorithms in a safe and controlled environment before hardware implementation. It facilitates rapid iteration, reduces the risk of hardware damage, and provides a cost-effective approach for exploring complex control strategies.
- We propose various RL-based methods aimed at improving controller performance, enabling the system to adapt to task-specific behaviors and optimize control strategies over time.
- A comprehensive ablation study of different RL algorithms is conducted on the simulated mobile robot manipulation task to assess the effectiveness of our methods. This analysis explores the influence of different approaches and parameters on task performance, providing valuable insights for future research.

[1]Mobile Industrial Robot project page

2 RELATED WORK

A Mobile Manipulator (MM) is a highly coupled system consisting of a manipulator arm attached to a mobile robot. This configuration contrasts with static manipulators, where the task space is constrained to a predefined configuration in a known space (Stilman, 2010). Controlling a dynamic platform such as an MM presents challenges, including motion planning and the management of redundancy across the entire system.

One extensively studied approach for controlling MMs is model-based control. For example, Model Predictive Control (MPC) has been applied to MMs (Minniti et al., 2021) to facilitate motion planning in unknown environments. To address the problem of additional degrees of freedom and dynamic obstacles, Wei et al. (Li and Xiong, 2019) proposed an optimization-based method for real-time obstacle avoidance. Their approach involved calculating the global robotic Jacobian matrix of the MM, followed by using MPC to plan control actions that resulted in calculated joint velocities for both the arm and the mobile base. Another promising technique is the combination of data-driven models with MPC to enhance MM performance. For instance, Carron et al. (Carron et al., 2019) used data gathered during platform movements to refine the model of the robotic arm and improve trajectory tracking performance. Their approach involved integrating inverse dynamics feedback linearization with a data-driven error model into the MPC framework.

Of particular relevance to our work is the Whole-Body Control (WBC) technique. For instance, Dietrich et al. (Dietrich et al., 2012) applied WBC using null space projection based on a dynamic model for redundancy resolution. Their algorithm enabled torque control of the robotic arm, which effectively scaled the apparent motor inertia, while simultaneously applying velocity commands in Cartesian coordinates. This method solved challenges like center-of-mass control, obstacle avoidance, and posture stabilization. Similarly, Tao et al. (Teng et al., 2021) employed WBC to enable efficient and complex grapevine pruning tasks using a non-holonomic MM.

To enhance task prioritization within the WBC framework, Kim et al. (Kim et al., 2019) introduced the Hierarchical Quadratic Programming (HQP) method. This approach facilitated the execution of complex tasks without causing discontinuities in the control input of the MM. Their controller was able to compute continuous control inputs while dynamically adjusting task priorities, utilizing a continuous task transition strategy. To improve generalization in uncertain environments and tackle complex tasks, recent research has increasingly focused on machine learning methods for MM operation (Lober et al., 2016; Welschehold et al., 2017; Wang et al., 2020b; Wang et al., 2020a; Jauhri et al., 2022). A prominent line of investigation has centered on using reinforcement learning (RL) to train MMs to perform complex behaviors by leveraging exploration and reward structures, without requiring explicit environment models. Research in this domain can be broadly divided into two categories.

The first category involves end-to-end RL approaches for solving WBC (Kindle et al., 2020; Wang et al., 2020b; Jauhri et al., 2022). One of the primary limitations of such methods is the requirement for vast amounts of data to learn an effective control policy. To address this limitation, the second category of research focuses on hybrid RL approaches that combine RL with traditional control methods (Jauhri et al., 2022; Iriondo et al., 2019; Iriondo et al., 2019). We explored this area of research as well. For example, Chalvatzaki et al. (Jauhri et al., 2022) developed a hybrid RL algorithm that integrates both discrete and continuous actions to facilitate the learning of a robust control policy. Their approach leveraged prior action probabilities from classical control methods derived from the operational robot workspace to enhance learning efficiency.

In addition to single-agent reinforcement learning (RL) settings, multi-agent RL has gained significant attention in recent years (Zhang et al., 2021). Researchers have focused on extending off-policy single-agent RL techniques to multi-agent environments, such as the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) (Lowe et al., 2020). Unlike in single-agent RL, each agent in a multi-agent system has limited access to the observations of other agents, which presents additional coordination and communication challenges. Furthermore, the high data requirements in RL often result in prolonged training periods. To address this, Offline RL (also known as Batch RL) has been proposed to leverage static datasets (Fujimoto et al., 2019) aiming to imitate the optimal behavior, thereby reducing the need for frequent interactions with the environment during training.

In this work, we aim to integrate RL techniques with the previously mentioned WBC mechanism, applied to a simulated mobile robot. Additionally, we present a comprehensive benchmark and ablation study of various RL algorithms to further validate and assess the effectiveness of our platform. Through this evaluation, we explore the strengths and limitations of different RL methods in mastering robotic behavior, providing valuable insights into their applicability for advanced robotic tasks.

3 PRELIMINARIES

Offline Reinforcement Learning vs Online Reinforcement Learning. Online Reinforcement Learning (RL) involves continuous interaction between the agent and the environment, with the training dataset being dynamically updated as new experiences are gathered. In contrast, Offline RL relies on a static dataset D, composed of transitions (O_t, A_t, R_t, O_{t+1}), where O_t represents the observations, A_t the actions, R_t the rewards, and O_{t+1} the subsequent observations. Since there is no interaction with the environment during the offline training process, a key challenge emerges: the distributional shift between the static dataset and the agent's evolving policy. This shift must be carefully managed to prevent degraded learning performance and ensure the effectiveness of the offline RL approach.

Offline Multi-Agent Reinforcement Learning. In multi-agent reinforcement learning (MARL), the objective is to learn an optimal joint policy that accounts for the interactions between all agents, rather than focusing solely on individual policies. Each agent, however, has access to only partial observations (O_t^i) of the environment. Under the Centralized Training with Decentralized Execution (CTDE) framework (Lowe et al., 2020), training is conducted using full-state information (S_t), while execution is based on each agent's local observations. Offline MARL (OMARL) extends the principles of MARL by leveraging static, pre-collected datasets (π_β generated) to learn optimal policies. In this setting, no further online interactions π_α are required.

4 METHODOLOGY

4.1 The Simulated Mobile Robot Platform

We develop a simulated mobile robot platform, named ***MirandaSim***, equipped with a robotic arm us-

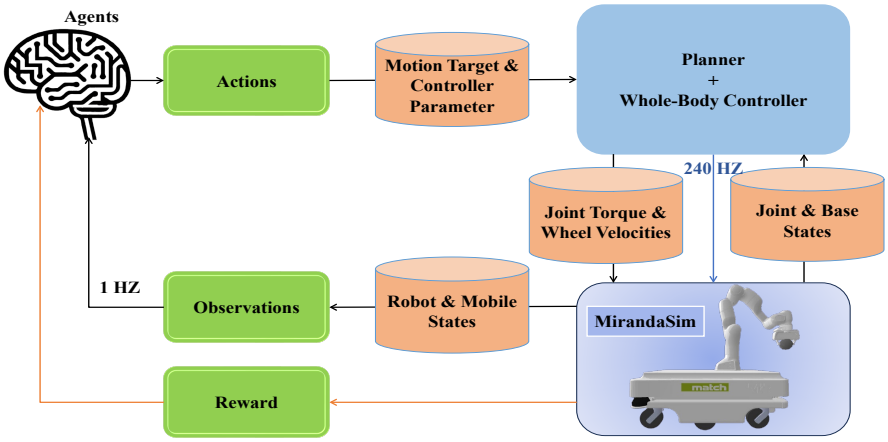
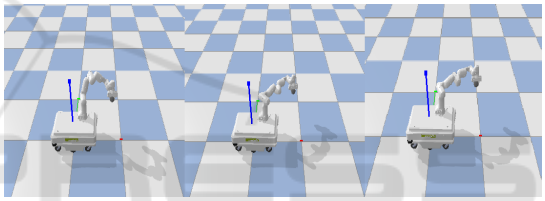


Figure 2: Mobile Industrial Robot mechanism. The platform called ***MirandaSim***, is developed using the Gym API and Pybullet Engine. The system operates at a frequency of 240 Hz for communication between the Whole-Body Control (WBC) planner and the simulated platform. Additionally, a 1 Hz frequency is employed to transmit observations to the proposed RL algorithms. In response, the RL module generates an action, which serves as the target for the WBC, and corresponding joint torque values are transmitted back to the platform, alongside feedback on the system's states. This structure enables the development of an efficient control block.

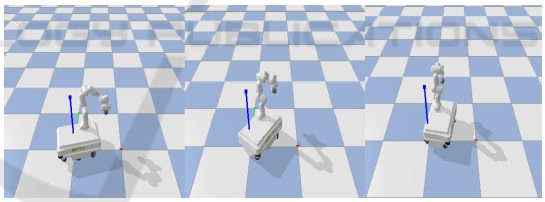
ing Pybullet and OpenAI Gym, as shown in Figure 2, which could be seen as an idealized model. This platform operates under WBC, which generates reliable movement trajectories by calculating joint torques and wheel velocities based on the arm's joint states and the mobile platform's base states. The communication between components occurs at a frequency of 240 Hz. Additionally, various RL techniques can be applied within this framework.

MirandaSim provides comprehensive state information, including the end-effector position of the robotic arm and the base position of the mobile platform. In this setup, RL determines the actions and supplies target points for the WBC. The reward function currently used is a dense reward, calculated as the negative distance between the achieved and desired goals.

The robot arm and the mobile platform can operate independently, following a random action policy. In Figure 3, we observe that the platform is capable of both movement and rotation. However, the previous motion planner and control system lacks sufficient coordination between the robot arm and the mobile platform. Both components move towards their individual goals separately, without synchronization. This highlights the need for a WBC technique (Teng et al., 2021), which could significantly enhance task performance in future research.



(a) The robot arm can move to a position in random test.



(b) The mobile platform is able to rotate to a specific angle in random test.

Figure 3: Random test of robot arm and mobile platform. The top graph shows the robot arm reaching a specific point without the platform moving, while the bottom graph illustrates the platform rotating to a specific angle while the robot arm remains stationary.

4.2 Whole-Body Controller

As outlined in (Kim et al., 2019), the kinematic and dynamic model of a nonholonomic mobile manipulator (MM) can be established, enabling the computation of control commands for task execution by solving a Quadratic Programming (QP) problem:

$$\begin{aligned} &\arg\min_{u} \ \|Au-b\|^2 \\ &\text{s.t. } \underline{d} \le Cu \le \overline{d} \end{aligned} \tag{1}$$

where u is the control input vector, e.g. the joint torques τ; A is the equivalent Jacobian matrix of the task; b is the reference value for task control and $\underline{d} \le Cu \le \overline{d}$ are equality and inequality constraints,

e.g. floating base dynamics, torque limits and moving platform related constraints.

To efficiently solve for redundancy and handle task conflicts, we use the Recursive Hierarchical Projection-Hierarchical Quadratic Programming (RHP-HQP) algorithm (Han et al., 2021), offering higher computational speed than the method described in (Kim et al., 2019). In an HQP with n levels, the task hierarchy at level k is solved in the projected space of higher-priority tasks:

$$u_k^* = u_{k-1}^* + P_{k-1} x_k^* \tag{2}$$

where u_k^* is the optimal control input at hierarchy k; P_{k-1} is the projection matrix constructed from a parameter matrix Ψ defining the task priorities and x_k^* is the optimal result of the k-th QP

$$\begin{aligned} &\arg\min_{x_k} \|A_k(P_{k-1}x_k + u_{k-1}^*) - b_k\|^2 \\ &\text{s.t. } \underline{d} \leq Cu \leq \overline{d} \end{aligned} \tag{3}$$

We define two sets of task priorities for MM, tailored to either optimize stability or reach performance, as shown in Table 1. The final priority matrix is determined using a parameter $\alpha \in [0,1]$, allowing for control of the trade-off between move stability of the whole system and reaching performance:

$$\Psi(\alpha) = \alpha\Psi_1 + (1-\alpha)\Psi_2 \tag{4}$$

By exploiting the parameter α, we can learn the trade-off between move stability and reaching performance.

Table 1: Two sets of predefined task priorities, Ψ_1 for better move stability and Ψ_2 for better manipulator's reaching. A lower number refers to a higher priority.

Task Description	Priority Level	
	Ψ_1	Ψ_2
Base velocity	1	3
Base orientation	1	3
Manipulator position	2	1
Manipulator orientation	3	2
Arm posture	4	4

4.3 DDPG Algorithm Based on Whole-Body Controller

The Deep Deterministic Policy Gradient (DDPG) algorithm, an Actor-Critic method, is widely recognized for its stability and efficiency in robot control tasks (Mnih et al., 2015). It utilizes an action-value function (critic) to guide the learning process and determines a deterministic policy $\pi(s) = a$ where the action a is generated by the actor-network. In our case, the DDPG algorithm is adapted to work in conjunction with the WBC.

The WBC has different task requirements, such as movement stability and manipulator accuracy. While achieving optimal performance in both is theoretically ideal, it is not feasible in real-time control scenarios. The DDPG algorithm helps balance these competing objectives in tasks like manipulator reaching.

The action space for the DDPG agent is represented by a 4×1 array, where the first three values correspond to the end-effector's position relative to the world frame, and the fourth value helps the agent learn balance through reward-based training.

To improve the efficiency of training, we incorporated Hindsight Experience Replay (HER) (Andrychowicz et al., 2017; Li et al., 2022). HER allows the agent to replay episodes with different goals by constructing hindsight goals from intermediate states, enabling faster learning from the achieved outcomes. Additionally, our platform supports offline data loading, inspired by methods in (Kalashnikov et al., 2018; Li et al., 2023b; Li et al., 2023a), where successful experiences are prioritized to accelerate learning. The complete DDPG-based algorithm for WBC as our online RL algorithm is presented in Algorithm 1.

Inputs: *Initialize main (critic, actor) and target (critic, actor) neural network weights and replay buffer*
Outputs: *Trained agent*
Initialize target network and main network weights and replay buffer;
while $i = 1, 2, ...T$ **do**
  Initialize the state;
  while $i = 1, 2, ...T$ **do**
    Load offline data (**Optional**);
    Generate *Action*;
    Obtain $State_{next}$, *Reward* from the planner based on **WBC**;
    Store the obtained information in replay buffer D;
    Compute the target value from a small batch in replay buffer (**HER techniques optional**);
    Update the main network critic by minimizing the error between the target value and critic Q;
    Update the main network actor using the sampled gradient;
    Update the target network;
  end
end

Algorithm 1: DDPG (Online RL algorithm) on Whole-Body Controller.

4.4 Offline RL Algorithm Based on Whole-Body Controller

The Twin Delayed Deep Deterministic Policy Gradient with Behavior Cloning (TD3+BC) algorithm (Fujimoto and Gu, 2021) serves as an effective reinforcement learning method applicable in batch reinforcement learning settings, demonstrating stable performance. When compared to other state-of-the-art offline reinforcement learning algorithms, TD3+BC significantly reduces overall running time costs. The algorithm consists of two main components. The first component is the traditional TD3 algorithm, which enhances the original DDPG approach by introducing an actor updating delay and employing a double Q-learning structure to mitigate the risk of overestimating Q-values during updates. Additionally, a Behavior Cloning (BC) term is incorporated, as illustrated by the following equation:

$$\mathcal{L}_{\mathrm{BC}} = \frac{1}{N}\sum_{i=1}^{N} \|a_i - \pi(s_i)\|^2 \tag{5}$$

Subsequently, the actor loss is augmented by the BC term, facilitating an update of the actor-network. The action space and other settings are the same as the previous DDPG algorithm.

As previously described, in our platform, the action space is defined by the first three values, which represent the end-effector position, along with the final control mode value. Given this, a multi-agent approach can effectively address this task. The first agent is tasked with determining the next state for the system, while the second agent focuses on maintaining the overall balance and stability of the platform.

Building on the TD3+BC framework, we propose a Multi-Agent Behavior Cloning (MABC) approach. Unlike standard TD3+BC, MABC simplifies the algorithm by considering only the difference between the actual action and the policy action, without incorporating a critic mechanism. This design allows for efficient offline RL learning while maintaining performance in control tasks. The complete offline RL algorithm—including single-agent TD3+BC, Multi-Agent TD3+BC (MATD3+BC), and MABC is outlined in Algorithm 2.

5 EXPERIMENT

The applications are implemented in a modular fashion utilizing Pybullet and OpenAI Gym, as illustrated in Fig. 1. OpenAI Gym serves as a robust toolkit for researching Deep Reinforcement Learning (DRL) algorithms. Additionally, the Pybullet engine offers excellent compatibility with the Gym, facilitating rapid simulation results. The RL baseline algorithm is sourced from Keras (Chollet et al., 2015) and other researchers' work (Pan et al., 2022), allowing for efficient integration into our simulation framework. Within this architecture, the RL agent determines the action strategy, while the WBC employs this strategy to execute motion planning and control.

In this section, we design one basic experiment that collectively forms a comprehensive task utilizing our simulated mobile robot platform. The robot is capable of grasping an object, subsequently, the platform, equipped with the robotic arm, maneuvers to a designated position while carrying the grasped mass. The DDPG algorithm is employed as an online RL solution with its own parameters. To achieve the task, we further investigate how offline RL and Offline Multi-Agent Reinforcement Learning (OMARL) contribute to enhancing performance.

5.1 Simulation Setup

The manipulator-reaching task follows the learning of the grasping object. In this phase, the action consists of adjusting the end-effector position and the parameter α, as previously mentioned. The observation encompasses the robot manipulator's position, the mobile platform's position, and the control mode value, which is designed to enhance performance by balancing accuracy and stability. The reward is formulated based on the error between the achieved state and the desired goal.

The desired goals are sampled based on specific intervals in the *xyz* space. The x sampling interval is defined as $(-0.8,-0.5)$ and $(1,1.5)$, accommodating both forward and backward movements. The y sampling interval is set to $(-0.2,0.2)$, while the z sampling interval is specified as $(0.55,0.8)$. The initial state is fixed with the joint values $[0,-0.215,0,-2.57,0,2.356,2.356,0.08,0.08]$. The dimensions of the actions and states for the neural networks are 4 and 10, respectively.

For multi-agent tasks, the setup involves two distinct agents with different roles and observations. The first agent is responsible for controlling the changes in the end-effector position, and its observation space consists of the state information necessary for guiding the robot's movement. The second agent, on the other hand, handles the parameter α. The observation for this agent is based on control mode data, allowing it to fine-tune the system's performance based on the dynamic requirements of the task.

Inputs: *Initialize main (critic, actor) and target (critic, actor) neural network weights and replay buffer*
Outputs: *Trained agent*
Initialize target network and main network weights and replay buffer;
while $i \leq T$ **do**
  Initialize the state;
  if *Single-agent Setup* **then**
    Load offline data from **WBC**; ▷ TD3+BC
    Compute the target value from a small batch in the replay buffer;
    Update the main network critic by choosing $Q_{\min} = \min(Q_1, Q_2)$;
    Update the main network actor with the BC loss term;
    Update the target network;
  else
    if *Use Critic Updating* **then**
      Load offline data from **WBC**; ▷ MATD3+BC
      for *Each Agent i* **do**
        Compute the target value from a small batch in the replay buffer;
        Update the main network critic for agent i by choosing $Q^i_{\min} = \min(Q^i_1, Q^i_2)$;
        Update the main network actor for agent i using the sampled gradient;
        Update the target network;
      end
    else
      Load offline data from **WBC**; ▷ MABC
      for *Each Agent i* **do**
        Compute the target value from a small batch in the replay buffer;
        Update the main network actor for agent i only with the BC loss term;
        Update the target network;
      end
    end
  end
end

Algorithm 2: Offline RL algorithm (single-agent and multi-agent) on Whole-Body Controller.

5.2 Network Structure

In this task, the DDPG algorithm, based on WBC, is implemented to learn how to reach the desired goal for the robot's end-effector. The details of the network hyperparameters are presented in Table 2.

Table 2: Network structure in manipulator reaching task.

Critic learning rate	0.001
Actor learning rate	0.001
Total epochs	100
Total episodes	20
Total steps per epoch	25
Standard deviation	0.1
Buffer capacity	5000
Buffer warm-up	5000
Batch size	256
τ	0.005
γ	0.95

For offline and multi-agent settings, the TD3+BC training information is as Table 3 shows.

Table 3: Offline RL and multi-agent RL Network structure in manipulator reaching task.

Critic learning rate	0.0005
Actor learning rate	0.0005
Total steps per epoch	25
Total train (parameter update) times	100000
Standard deviation	0.1
Buffer capacity	30000
Batch size	256
τ	0.005
γ	0.95

5.3 Reward Mechanism

In this scenario, the reward mechanism consists of three components. The first component penalizes the agent if the end-effector does not reach the goal position. If the goal position is successfully reached, a positive reward is provided to encourage successful

behavior, as shown in Eq. 6.

$$r_1(s,a) = \begin{cases} -d & \text{, if not reaching} \\ +10 & \text{, else} \end{cases} \quad (6)$$

Here, d represents the distance between the current robot manipulator and the desired goal.

Additionally, we observe that in some cases, both the mobile platform and manipulator may not reach the target position in a few small steps. Therefore, we compute the difference between the target and achieved positions for both the manipulator and the mobile platform. Let d_{diffmm} denote the difference between the target and reached manipulator positions, and d_{diffmp} denote the difference between the target and reached mobile platform base positions. The corresponding reward components are defined as shown in Eq. 7 and Eq. 8.

$$r_2(s,a) = -5 \times d_{diffmm} \quad (7)$$

$$r_3(s,a) = -5 \times d_{diffmp} \quad (8)$$

Finally, the reward that we used to learn is the sum of the above three rewards as Eq.9.

$$r(s,a) = r_1(s,a) + r_2(s,a) + r_3(s,a) \quad (9)$$

5.4 Results

In this section, learning curves of tasks and performance are demonstrated. The learning curves are sampled multiple times (three random seeds). We present the mean value and standard deviation area of these learning curves. Based on that, the task performance is analyzed with the trained agent.

5.4.1 Online Train

The first experiment is the manipulator reaching task using online RL techniques. In this experiment, both the vanilla DDPG algorithm and the DDPG enhanced with the Hindsight Experience Replay (HER) technique are employed to learn the task behavior. The results are presented in Fig. 4 and Fig. 5.

This task proves to be more challenging than the initial grasping task, as the agent must learn not only the action policy but also the parameter α to balance accuracy and stability. The HER technique significantly enhances both training efficiency and success rate. In Fig. 4, the green curves represent the performance with HER, while the red curves show the results without HER. The comparison is notable, with the HER-enhanced agent achieving a success rate of 70% by approximately the 90th episode, whereas the vanilla DDPG reaches under 40%.

We then evaluate the reference performance of the trained agent. As shown in Fig. 6, the mobile platform successfully reaches both positions in front of

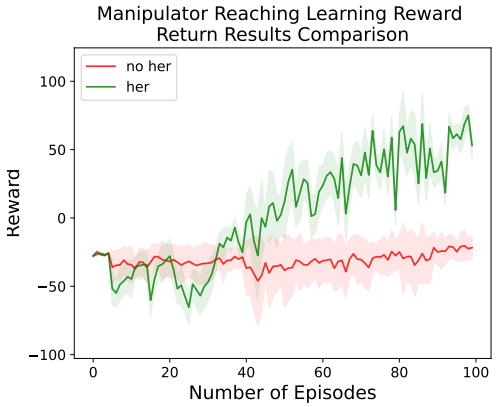


Figure 4: Reward comparison in the manipulator reaching task. The shaded area is the standard deviation.

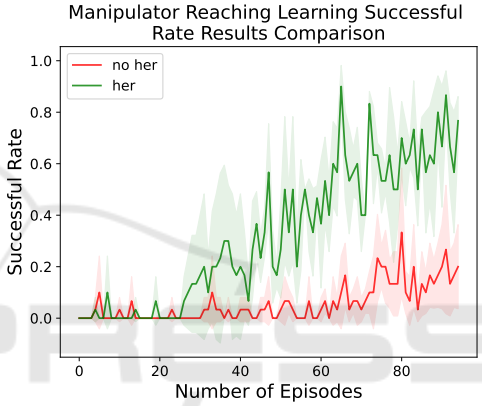


Figure 5: Successful rate comparison in Manipulator reaching task. The shaded area is the standard deviation.

and behind its initial location during the end-effector reaching task. Notably, forward movements require fewer steps than backward movements in our simulated platform. In multi-agent settings, online training is often time-consuming and yields fewer successful examples. Therefore, this paper does not present multi-agent online training. Instead, successful multi-agent data is derived from the single-agent successful dataset, as discussed in a subsequent section.

5.4.2 Offline Train

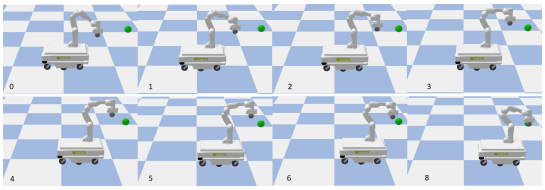
As previously discussed, offline reinforcement learning (RL) techniques depend on a static dataset to accomplish specific tasks. Similar to supervised learning, the agent can leverage prior knowledge from the pre-collected dataset. Consequently, the quality of the offline dataset is critical for effective offline reinforcement learning. Therefore, we not only implement the proposed algorithms but also conduct an ablation study examining various datasets and hyperparameters in the neural network structure.

Table 4: Performance comparison of Offline RL with different datasets in the manipulator reaching task, the Successful Rate is collected at the iteration times of 18×10^4.

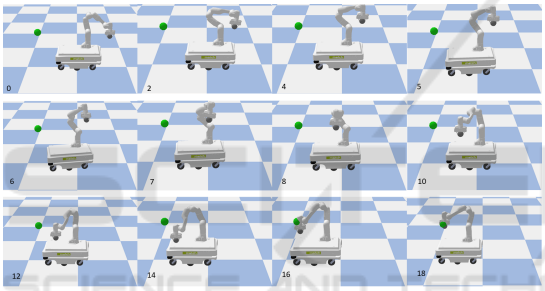
Dataset	Successful Rate (%)
60k good data	26.00($\pm$14.42)
30k good data	**34.00**($\pm$**32.19**)
5k random + 5k medium + 20k good data	12.67($\pm$15.53)
10k random + 20k good data	16.00($\pm$17.09)

Table 5: Performance comparison of Offline MARL with different datasets in the manipulator reaching task.

Dataset	Successful Rate (%)
60k good data (Only MABC)	**78**($\pm$**7.21**)



(a) Forward performance in the end-effector reaching task



(b) Backward performance in the end-effector reaching task

Figure 6: Reference performance in the manipulator reaching task. Subfigure (a) illustrates the forward movements, while subfigure (b) shows the backward movements. In each sub-figure, the current step in an episode is displayed in the bottom left corner. The maximum number of steps per episode in our setup is 25.

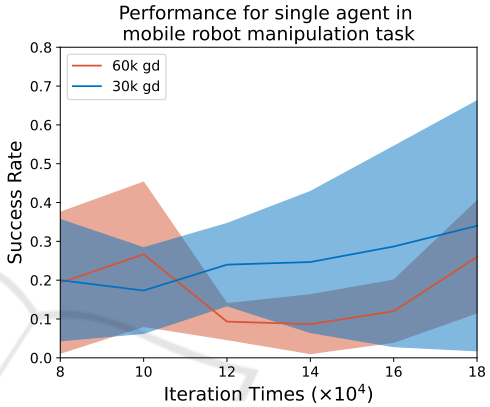


Figure 7: Successful rate comparison in Manipulator reaching task for different amounts of good data. Here, 60k ***gd*** and 30k ***gd*** mean 60 and 30 thousand good data records, respectively. The solid area represents the standard deviation.

Single-Agent. In our study, we initially collect two datasets comprising 30,000 and 60,000 steps, respectively. These datasets are derived from the rollout of a pre-trained agent that successfully executed the manipulator-reaching task. For the sake of clarity, we refer to this as *good* data (***gd***). We then examine whether an increase in data quantity could enhance robot manipulation performance. As illustrated in Figure 7 and summarized in Table 4, we find that a larger dataset does not necessarily correlate with improved learning performance. However, it does contribute to greater stability in performance, as evidenced by the deviation rates of 14.42% for the larger dataset compared to 32.19% for the smaller one.

To our knowledge, the term "pre-trained agent" refers to substantial prior work. Specifically, achieving optimal agent-generated rollouts necessitates comprehensive online training. In this context, we propose employing the Proportional-Derivative (PD) method to collect successful trajectories based on the WBC framework. These trajectories will subsequently be stored as offline data. However, we encounter challenges with the PD method in executing backward movements within the 25-step constraint of our platform's WBC. To address this, we extend the episode length to 50 steps and tested the agent's performance under this modified configuration, while the pre-trained agent continued to operate within the original 25-step framework. As illustrated in Figure 8, we observe that the PD method yields results comparable to the complete set of *good* data when used as offline data.

Further, using only successful data can limit the agent's learning, as it will never encounter or learn how to recover from failure scenarios. To address this, we collect a second dataset consisting of 20,000 successful states from the pre-trained agent and an additional 10,000 states generated randomly(***rd***). Moreover, the third dataset is created with 20,000 success-

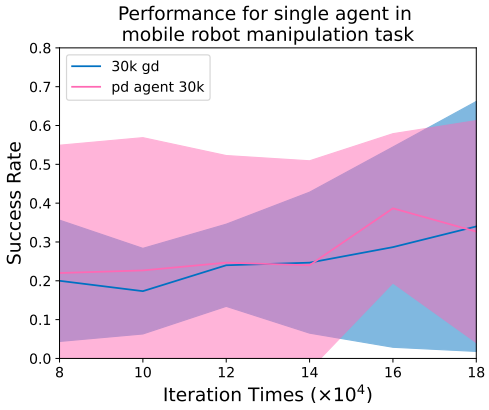


Figure 8: Successful rate comparison in Manipulator reaching task for data generated by different agents. Here ***pd*** means that the data is collected by a PD(proportional-derivative) agent, ***gd*** means good data record. The shaded area is generated by computing the standard deviation from the training results.

ful data (***gd***), 5,000 random data (***rd***), and medium data (***md***) which created by first 10 random steps and 15 steps at most generated by pre-trained agent in the whole 25 steps of one episode. In theory, this diverse dataset provides a broader distribution of states, allowing the agent to learn how to handle both success and failure scenarios. From Figure 9, the results show that all ***gd*** generated offline data perform the best whether on successful rate or the standard deviation. The TD3+BC algorithm is used to train the agents on both datasets for 180,000 steps. The full results are shown in Table 4.

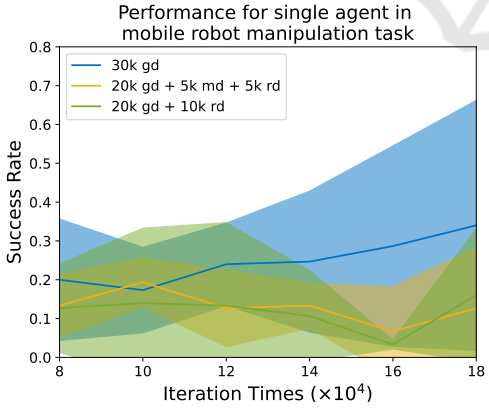


Figure 9: Successful rate comparison in Manipulator reaching task for different data composition, where ***gd*** means good data records, ***md*** and ***rd*** for medium and random data records, respectively. The solid curve is the mean successful value of the training process. The filled region with the responding color is the standard deviation area.

From Table 4, we notice that a more diverse dataset, including a mixture of successful and random trajectories, results in a lower success rate compared to using only successful data. In our view, the underlying reason is the limited size of the dataset combined with a high proportion of random data. The importance of data distribution in offline RL is still valuable.

For hyperparameter tuning, we conduct experiments by training offline RL agents with varying batch sizes. As shown in Figure 10, the larger batch sizes result in a lower success rate for the reaching task (1024 batch size with 9.33% at 180,000 steps). For batch size 512, the successful rate is 10.67%. This suggests that simply increasing hyperparameter values does not always lead to better performance.

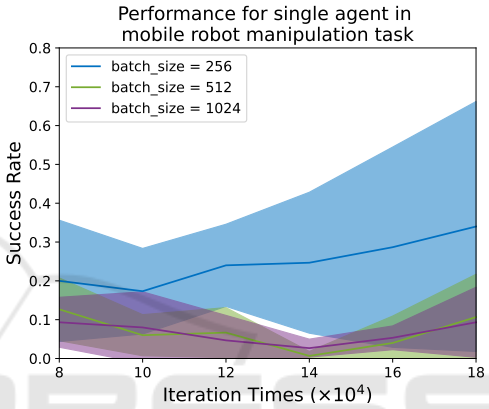


Figure 10: Successful rate comparison in Manipulator reaching task for different batch size settings. The solid curves represent the mean values over three random seeds. The shaded region is the standard deviation.

Multi-Agent. Furthermore, the performance of multi-agent settings is evaluated using the 30k successful data previously collected in single-agent configurations. The results are summarized in Table 5. Interestingly, the simple MABC algorithm exceeded our expectations, achieving a mean success rate of 78% with a relatively low standard deviation. However, despite this promising result, the overall success rate remains below the desired level, and MATD3+BC, in particular, failed to yield satisfactory outcomes during training. This highlights the need for further optimizations, such as expanding the dataset and incorporating more diverse data distributions, to enhance performance and stability.

6 CONCLUSION

In this paper, we present the development of a simulated mobile robot with an arm platform, built with Pybullet and Gym, which supports a reinforcement learning framework. Instead of relying on traditional controllers and planners, we employ a Whole-Body

Controller to provide low-level control. On top of this, reinforcement learning is integrated to enhance the controller's overall performance. To evaluate the effectiveness of the platform and the proposed methods, we conducted one typical simulated experiment. The results demonstrate the feasibility of the platform and its associated methodologies. Furthermore, the system exhibits potential for handling more complex tasks in the future.

In our future work, we aim to extend this research to real-world experiments based on the simulated results. Additionally, further investigation is needed into multi-agent training with critic updates.

REFERENCES

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. (2017). Hindsight experience replay. *arXiv preprint arXiv:1707.01495*.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym.

Carron, A., Arcari, E., Wermelinger, M., Hewing, L., Hutter, M., and Zeilinger, M. N. (2019). Data-driven model predictive control for trajectory tracking with a robotic arm. *IEEE Robotics and Automation Letters*, 4:3758–3765.

Chollet, F. et al. (2015). Keras.

Coumans, E. and Bai, Y. (2016–2021). Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org.

Dietrich, A., Wimböck, T., Albu-Schäffer, A. O., and Hirzinger, G. (2012). Reactive whole-body control: Dynamic mobile manipulation using a large number of actuated degrees of freedom. *IEEE Robotics & Automation Magazine*, 19:20–33.

Fujimoto, S. and Gu, S. S. (2021). A minimalist approach to offline reinforcement learning. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 20132–20145. Curran Associates, Inc.

Fujimoto, S., Meger, D., and Precup, D. (2019). Off-policy deep reinforcement learning without exploration.

Grabowski, A., Jankowski, J., and Wodzyński, M. (2021). Teleoperated mobile robot with two arms: the influence of a human-machine interface, vr training and operator age. *International Journal of Human-Computer Studies*, 156:102707.

Guo, H., Su, K.-L., Hsia, K.-H., and Wang, J.-T. (2016). Development of the mobile robot with a robot arm. In *2016 IEEE International Conference on Industrial Technology (ICIT)*, pages 1648–1653.

Guo, N., Li, C., Wang, D., Song, Y., Liu, G., and Gao, T. (2021). Local path planning of mobile robot based on long short-term memory neural network. *Automatic Control and Computer Sciences*, 55:53–65.

Han, G., Wang, J., Ju, X., and Zhao, M. (2021). Recursive hierarchical projection for whole-body control with task priority transition. *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11312–11319.

Iriondo, A., Lazkano, E., Susperregi, L., Urain, J., Fernandez, A., and Molina, J. (2019). Pick and place operations in logistics using a mobile manipulator controlled with deep reinforcement learning. 9(2):348.

Jauhri, S., Peters, J., and Chalvatzaki, G. (2022). Robot learning of mobile manipulation with reachability behavior priors. *IEEE Robotics and Automation Letters*, 7:8399–8406.

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., et al. (2018). Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*.

Kim, S., Jang, K., Park, S., Lee, Y., Lee, S. Y., and Park, J. (2019). Whole-body control of non-holonomic mobile manipulator based on hierarchical quadratic programming and continuous task transition. *2019 IEEE 4th International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 414–419.

Kindle, J., Furrer, F., Novkovic, T., Chung, J. J., Siegwart, R., and Nieto, J. (2020). Whole-body control of a mobile manipulator using end-to-end reinforcement learning. *arXiv preprint arXiv:2003.02637*.

Li, C., Liu, Y., Bing, Z., Schreier, F., Seyler, J., and Eivazi, S. (2023a). Accelerate training of reinforcement learning agent by utilization of current and previous experience. In *ICAART (3)*, pages 698–705.

Li, C., Liu, Y., Bing, Z., Seyler, J., and Eivazi, S. (2022). Correction to: A novel reinforcement learning sampling method without additional environment feedback in hindsight experience replay. In Kim, J., Englot, B., Park, H.-W., Choi, H.-L., Myung, H., Kim, J., and Kim, J.-H., editors, *Robot Intelligence Technology and Applications 6*, pages C1–C1, Cham. Springer International Publishing.

Li, C., Liu, Y., Hu, Y., Schreier, F., Seyler, J., and Eivazi, S. (2023b). Novel methods inspired by reinforcement learning actor-critic mechanism for eye-in-hand calibration in robotics. In *2023 IEEE International Conference on Development and Learning (ICDL)*, pages 87–92.

Li, W. and Xiong, R. (2019). Dynamical obstacle avoidance of task- constrained mobile manipulation using model predictive control. *IEEE Access*, 7:88301–88311.

Lober, R., Padois, V., and Sigaud, O. (2016). Efficient reinforcement learning for humanoid whole-body control. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 684–689. IEEE.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. (2020). Multi-agent actor-critic for mixed cooperative-competitive environments.

Minniti, M. V., Grandia, R., Fäh, K., Farshidian, F., and Hutter, M. (2021). Model predictive robot-environment interaction control for mobile manipulation tasks. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1651–1657.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Pan, L., Huang, L., Ma, T., and Xu, H. (2022). Plan better amid conservatism: Offline multi-agent reinforcement learning with actor rectification. In *International Conference on Machine Learning*.

Ramalepa, L. P. and Jr., R. S. J. (2021). A review on cooperative robotic arms with mobile or drones bases. *Machine Intelligence Research*, 18(4):536–555.

Stilman, M. (2010). Global manipulation planning in robot joint space with task constraints. *IEEE Transactions on Robotics*, 26(3):576–584.

Teng, T., Fernandes, M., Gatti, M., Poni, S., Semini, C., Caldwell, D. G., and Chen, F. (2021). Whole-body control on non-holonomic mobile manipulation for grapevine winter pruning automation *. *2021 6th IEEE International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 37–42.

Tinós, R., Terra, M. H., and Ishihara, J. Y. (2006). Motion and force control of cooperative robotic manipulators with passive joints. *IEEE Transactions on Control Systems Technology*, 14(4):725–734.

Uehara, H., Higa, H., and Soken, T. (2010). A mobile robotic arm for people with severe disabilities. In *2010 3rd IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 126–129.

Wang, C., Zhang, Q., Tian, Q., Li, S., Wang, X., Lane, D., Pétillot, Y. R., Hong, Z., and Wang, S. (2020a). Multi-task reinforcement learning based mobile manipulation control for dynamic object tracking and grasping. *2022 7th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*, pages 34–40.

Wang, C., Zhang, Q., Tian, Q., Li, S., Wang, X., Lane, D., Pétillot, Y. R., and Wang, S. (2020b). Learning mobile manipulation through deep reinforcement learning. *Sensors (Basel, Switzerland)*, 20.

Welschehold, T., Dornhege, C., and Burgard, W. (2017). Learning mobile manipulation actions from human demonstrations. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3196–3201.

Xin, J., Zhao, H., Liu, D., and Li, M. (2017). Application of deep reinforcement learning in mobile robot path planning. In *2017 Chinese Automation Congress (CAC)*, pages 7112–7116. IEEE.

Zhang, K., Yang, Z., and Başar, T. (2021). Multi-agent reinforcement learning: A selective overview of theories and algorithms.