IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Pre-Grasp Approaching on Mobile Robots: A Pre-Active Layered Approach

Lakshadeep Naik, Sinan Kalkan, and Norbert Krüger

***Abstract*—In Mobile Manipulation (MM), navigation and manipulation are generally solved as subsequent disjoint tasks. Combined optimization of navigation and manipulation costs can improve the time efficiency of MM. However, this is challenging as precise object pose estimates, which are necessary for such combined optimization, are often not available until the later stages of MM. Moreover, optimizing navigation and manipulation costs with conventional planning methods using uncertain object pose estimates can lead to failures and hence requires re-planning. Hence, in the presence of object pose uncertainty, pre-active approaches are preferred. We propose such a pre-active approach for determining the base pose and pre-grasp manipulator configuration to improve the time efficiency of MM. We devise a Reinforcement Learning (RL) based solution that learns suitable base poses for grasping and pre-grasp manipulator configurations using layered learning that guides exploration and enables sample-efficient learning. Further, we accelerate learning of pre-grasp manipulator configurations by providing dense rewards using a predictor network trained on previously learned base poses for grasping. Our experiments validate that in the presence of uncertain object pose estimates, the proposed approach results in reduced execution time. Finally, we show that our policy learned in simulation can be easily transferred to a real robot.**

***Index Terms*—Mobile manipulation, reinforcement learning.**

I. Introduction

MOBILE Manipulation (MM) has been generally addressed by sequentially solving navigation and manipulation tasks [1], [2]. The sequential approach is time inefficient as navigation and manipulation are treated as two separate problems. This is a hindrance for a widespread use of mobile manipulators as there are often hard time constraints on the execution time of the task at hand.

Manuscript received 30 September 2023; accepted 8 January 2024. Date of publication 24 January 2024; date of current version 5 February 2024. This letter was recommended for publication by Associate Editor D. Papageorgiou and Editor M. Vincze upon evaluation of the reviewers' comments. This work was supported in part by the Innovation Fund Denmark (in the context of the FacilityCobot project), and in part by the European Union's Horizon 2020 project Fluently under Grant Agreement 958417. *(Corresponding author: Lakshadeep Naik.)*

Lakshadeep Naik is with SDU Robotics, Faculty of Engineering, University of Southern Denmark, 5230 Odense, Denmark (e-mail: lana@mmmi.sdu.dk).

Sinan Kalkan is with the Department of Computer Engineering, Middle East Technical University, 06531 Ankara, Türkiye (e-mail: skalkan@metu.edu.tr).

Norbert Krüger is with SDU Robotics, Faculty of Engineering, University of Southern Denmark, 5230 Odense, Denmark, and also with the Danish Institute for Advanced Studies (DIAS), 5230 Odense, Denmark (e-mail: norbert@mmmi.sdu.dk).

The code repository and the supplementary video can be found on the project webpage https://lakshadeep.github.io/pgamr/.

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2024.3358077, provided by the authors.

Digital Object Identifier 10.1109/LRA.2024.3358077

The time efficiency of MM can be improved by jointly solving navigation and manipulation tasks. For example, by using whole-body (base & manipulator) motion control [3], [4], [5], [6], [7], determining optimal base pose for grasping [8], [9], [10], or by jointly planning base pose and pre-grasp manipulator configuration [11], [12]. The whole-body motion control methods attempt to directly reach the desired End-Effector (EE) pose by combining base & manipulator motion. They do not explicitly plan the base pose for grasping.

Existing works have used model-based control or learning for whole-body motion control. The model-based methods make certain assumptions regarding the kinematic model of the robot [5], [13]. This makes it difficult to realize transfers to robots with different embodiments. On the other hand, the learning-based methods can be applied to different robots as they do not make any assumptions regarding the robot's kinematic model. However, due to the complexity of learning whole-body motion, an error of a few centimeters in reaching the desired EE pose is usually considered a success [3], [5], [7]. This makes them unsuitable for applications such as grasping where the margin of error is small and failures can be expensive.

Hence, for tasks such as robotic grasping, explicit base placement along with a suitable pre-grasp manipulator configuration is preferred, i.e., the robot first identifies the suitable base pose and pre-grasp manipulator configuration, then navigates to this base pose while adopting the pre-grasp manipulator configuration and finally performs the grasp only moving the manipulator [11]. It is assumed that high manipulability (i.e. robot's ability to easily reach and move its end-effector to different positions and orientations in the task space) can effectively compensate for any errors in the robot's self-localization and initial object pose estimate used for planning. However, in practice, there is considerable uncertainty in object pose estimation, particularly when the robot is far away from the object [14], [15]. Hence, planning the base pose and pre-grasp manipulator configuration using uncertain object pose estimates often leads to failures and requires re-planning.

In this work, we propose a pre-reactive approach for selecting an optimal base pose for grasping and a pre-grasp manipulator configuration such that the overall time cost of MM is optimized. Using search-based methods such as Inverse Reachability Map (IRMs) [9], [10] for finding an optimal base pose and pre-grasp manipulator configuration in a reactive setup is computationally expensive. Previous works [8], [15] have successfully used learning-based methods for learning base poses for grasping. Motivated by these successes, we also propose a learning-based approach.

Reinforcement learning (RL) is a promising approach since it can learn by interacting with the environment. However, it requires a large amount of training data (sample inefficiency),

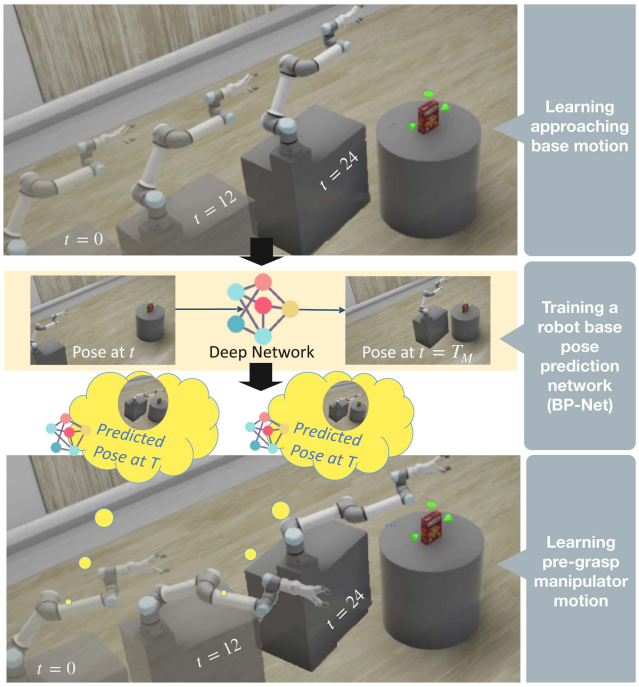

Fig. 1. First row: The robot learns a policy to approach the target object for grasping with its base motion aiming to reduce navigation time & ensure the availability of Inverse Kinematics solutions to the grasp pose. Second row: *Base pose prediction network* is trained using experience from previously learned policies. Third row: The robot learns the *pre-grasp manipulator motion* policy aiming to reduce grasp execution time by following previously learned policies and intermediate rewards provided by the BP-Net.

especially in scenarios with large action spaces. To address this challenge, prior knowledge can be leveraged to guide exploration and improve sample efficiency [16]. For example, in [8], first, a policy is learned to solve a simplified task and then the learned policy is used as a prior for learning more complex tasks.

The action space in our problem includes the base pose and the pre-grasp manipulator configuration. We address exploration and sample efficiency issues due to large action space by breaking the problem down into several hierarchical learning layers such that each layer facilitates the learning of the next [17]. To enable layered learning, we formulate the problem as predicting relative base and manipulator motion and making a discrete decision if the grasp should be performed in the resulting state (i.e., base pose and manipulator configuration), instead of directly learning the base pose and pre-grasp manipulator configuration. Fig. 1 describes our approach.

First, we learn an *approaching base motion* policy to approach the object for grasping using its base motion aiming to minimize the navigation time (see Fig. 1 first row). This is followed by learning the *grasp decision* policy that learns if a grasp should be attempted from the current base pose (while following the *approaching base motion* policy) with the aim to maximize the number of Inverse Kinematics (IK) solutions to the grasp poses.

Since suitable pre-grasp manipulator configuration depends on the selected base pose for grasping, this knowledge can be used to accelerate learning. We achieve this by training a *base pose prediction* network (BP-Net) that learns to predict a suitable base pose for grasping given an arbitrary starting pose using experience from the already learned *approaching base motion* & *grasp decision* policies (see Fig. 1 second row). Finally, the *pre-grasp manipulator motion* policy is learned (while following previously learned policies and using BP-Net for providing intermediate rewards to accelerate learning) with the goal of attaining the suitable pre-grasp manipulator configuration that reduces grasp execution time (see Fig. 1 third row).

We make the following main contributions:

- We formulate the problem of improving the time efficiency of MM as an RL problem and sequentially learn the *approaching base motion*, *grasp decision* & *pre-grasp manipulator motion* policies using layered learning to effectively guide exploration and enable sample-efficient learning.
- We propose a *robot base pose prediction network* (BP-Net), to accelerate learning of *pre-grasp manipulator motion* policies. This network provides dense rewards using the experience from previously learned *approaching base motion* and *grasp decision* policies.
- We validate our approach by comparing it against several baselines that explicitly plan robot base pose for grasping in a simulated environment.
- We successfully transfer the learned policy from simulation to a real robot.

II. Related Work

A. Explicit Base Placement Planning in MM

Determining the appropriate base pose for grasping often involves using Inverse Reachability Maps (IRM) [9], [10]. However, the use of IRM to plan a suitable base pose is computationally intensive due to the 6D search space. Consequently, recent work [8] has introduced a learning-based approach to predict base poses for grasping. In addition to planning the base pose for grasping, other works [11], [12] also plan manipulator configuration.

The object pose used for computing the base pose and the robot's self-localization have an uncertainty associated with them. The aforementioned works ignore these uncertainties. Very few works take these uncertainties into account. Meng et al. [18] have tried to address uncertainty in the robot's self-localization by assuming Gaussian uncertainty in the robot's localization while determining the base pose using IRM. In [15], action-related places are proposed that take into account take into account both the uncertainty in the object pose and the robot's localization. However, they consider very simplified action i.e. just the robot base position with a fixed orientation.

We address uncertainty in both the object pose estimate and the robot's self-localization by proposing a pre-active approach for planning the base pose and pre-grasp manipulator configuration.

B. Transfer Learning (TL) in RL

TL in RL involves first training the agent on one or more source tasks and use the knowledge acquired in the source task to aid in solving the target task [16]. Curriculum learning (CL) [19] is the form of TL wherein experience acquired by the agent over time is sorted in order to accelerate learning. As in RL, the entire set of samples is not available ahead of time, CL involves ordering the tasks such that the behavior policy learned is useful for acquiring good samples in future tasks [16]. Layered Learning (LL) is another form of TL wherein a complex task is hierarchically decomposed into sub-tasks. Each sub-task is

separately learned and directly facilitates the learning of the next higher sub-task [17].

In our work, we deal with a task that involves learning different actions (base motion, manipulator motion, and grasp decision). These actions are dependent on each other. For example, only when the robot has learned *approaching base motion* to reach base pose for grasping, it can learn grasp decision (i.e. when to grasp). Further, only when the robot has learned to reach a suitable base pose and grasp decision, it can learn pre-grasp manipulator configuration that reduces grasp execution time by using *pre-grasp manipulator motion*. LL allows such hierarchical decomposition of a complex task into sub-tasks to learn each sub-task separately while directly facilitating the learning of the next sub-task in the hierarchy.

C. Reward Design in RL

Reward design is crucial in RL as it is responsible for determining the agent's behavior. Since extrinsic rewards provided by the environment to the agent are often sparse, additional dense intermediate rewards are commonly introduced to accelerate learning [20]. Reward shaping involves providing the learning agent with extra shaped rewards to guide exploration and expedite learning [21], [22]. Another type of dense intermediate reward used in RL is intrinsic reward, which aims to incentivize exploration by encouraging the agent to explore unexplored regions of the exploration space [23], [24], [25]. To this end, approximator networks such as predictor network [24] or Intrinsic Curiosity Module [25] have been proposed to provide intrinsic rewards.

In this work, we use a predictor network for providing shaped rewards. We propose a BP-Net that uses experience from the already learned *approaching base motion* policy and provides a dense reward to indicate relevant exploration spaces for learning *pre-grasp manipulator motion* policy.

III. Problem Statement

We assume that tasks within MM, including object selection, grasp pose sampling, pre-grasp approaching, and grasping (trajectory planning, execution and gripper activation), are addressed independently. We focus on pre-grasp approaching to achieve a time-efficient grasp. Additionally, we assume that the workspace is characterized by a convex shape (for example, tables with rectangular or circular shapes).

We formulate this as an RL task. During learning, the object is positioned at the center of a cylindrical table with a diameter equal to the maximum reach of the manipulator. Each learning episode consists of the robot approaching the target object for grasping using both its base and manipulator motion. The objective of the RL agent is to reach the base pose for grasping while also attaining a pre-grasp manipulator configuration such that the time cost of MM is minimized.

Learning such a policy requires information about the object pose in the robot frame, the manipulator state (joint configuration) as well as actions for moving the robot's base, manipulator joints, and making a decision about when to attempt a grasp. Since shoulder and elbow joints have lower velocities and accelerations compared to wrist joints, their motion significantly impacts the grasp execution time. Hence, during the pre-grasp motion, we only learn the motion of the shoulder and elbow joints.

Thus, the state space consists of $\mathcal{S} = \{s_{\text{obj}}, s_{\text{arm}}\}$ where $s_{\text{obj}} \in \text{SE}(3)$ is the object pose in the robot base frame and $s_{\text{arm}} \in \mathbb{R}^3$ contains the shoulder and elbow joint positions of the manipulator. We use $\mathcal{S}^{\text{grasp}} \subset \mathcal{S}$ to denote the state where the robot decides to attempt a grasp when the episode ends. The action space consists of three actions $\mathcal{A} = \{a_{\text{base}}, a_{\text{arm}}, a_{\text{grasp}}\}$ where $a_{\text{base}} \in \mathbb{R}^2$ is the base linear and angular velocity, $a_{\text{arm}} \in \mathbb{R}^3$ denotes the rotation commands for robot's shoulder and elbow joints, and $a_{\text{grasp}} \in \{True, False\}$ is the binary decision to grasp.

An episode ends when the agent surpasses a predefined maximum number of time steps, collides with the table, or chooses to execute the grasp using the a_{grasp} action. All states and actions are internally represented with a time variable t, which, however, is dropped in our notations for convenience.

IV. Proposed Approach

In this section, we present our proposed approach, which utilizes Layered Learning (LL) [17] to learn the desired policy. First, we describe the task decomposition into sub-tasks and the sequence in which these sub-tasks are learned within the LL framework (Section IV-A). Next, we outline the learning process using separate sequential policies (Section IV-B). Finally, in Section IV-C, we introduce the BP-Net, which provides dense rewards to accelerate the learning process.

A. Task Decomposition and Learning Sequence

We decompose the main task into three sub-tasks, each aiming to learn individual actions $a_{\text{base}}, a_{\text{arm}}, a_{\text{grasp}}$ using separate policies $\pi_{\text{base}}, \pi_{\text{arm}}, \pi_{\text{grasp}}$. The *approaching base motion* policy π_{base} is responsible for learning action a_{base}, while the *pre-grasp manipulator motion* policy learns action a_{arm}, and the *grasp decision* policy is tasked with learning action a_{grasp}. All three policies are learned using the Actor-Critic algorithm [20]. $Q^{\pi_{\text{base}}}$, $Q^{\pi_{\text{arm}}}$ and $Q^{\pi_{\text{grasp}}}$ are the corresponding Q-functions for policies $\pi_{\text{base}}, \pi_{\text{arm}}$ and π_{grasp} respectively. As shown in Fig. 2, these policies are learned in the sequence: *approaching base motion* (Layer 1), *grasp decision* (Layer 2) and *pre-grasp manipulator motion* (Layer 3).

The *grasp decision* π_{grasp} policy is learned before learning the *pre-grasp manipulator motion* π_{arm} policy as the availability of Inverse Kinematics (IK) solutions to reach the grasp poses depends on the base pose from where grasp is attempted. This formulation also allows us to provide dense rewards to accelerate learning of the *pre-grasp manipulator motion* policy in Layer 3 using the learned base poses for grasping in Layers 1 and 2 (as described in Section IV-C).

Further, instead of learning the *approaching base motion* and *grasp decision* with a single policy, we learn them using two distinct policies. The reason for this separation is that the grasp action a_{grasp} is an episode-ending action and prevents exploration for learning action a_{base}. By sequentially learning the *approaching base motion* policy π_{base} first and then the *grasp decision* policy π_{grasp}, we achieve sample-efficient learning.

B. Learning With Separate Sequential Policies

Layer 1: In Layer 1, base motion action a_{base} is learned using policy π_{base} (see Fig. 2(a)):

$$a_{\text{base}} \sim \pi_{\text{base}}(\,\cdot\,|s_{\text{obj}};\ \phi_{\text{base}}), \tag{1}$$

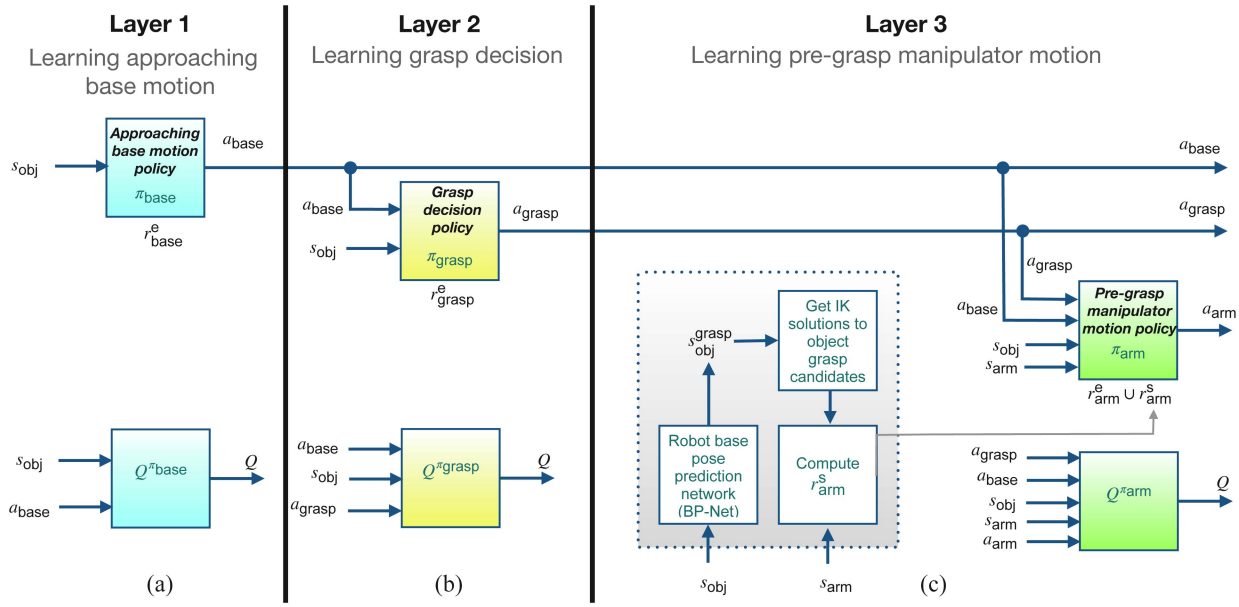

Fig. 2. Proposed layered approach: (a) Layer 1: Learning to approach the object for grasping (base motion) (b) Layer 2: Learning to determine when to attempt grasp (c) Layer 3: Learning the pre-grasp manipulator motion.

where ϕ_{base} are learnable parameters. The extrinsic reward is defined as follows:

$$\begin{aligned} r^{\text{e}}_{\text{base}}(s_{\text{obj}}, a_{\text{base}}) = \alpha_1 \cdot \Delta d_{\text{obj}}(s_{\text{obj}}, a_{\text{base}}) \\ + \alpha_2 \cdot \mathbf{1}(\text{collision}(s_{\text{obj}}, a_{\text{base}})) \\ + \alpha_3 \cdot |\text{IK}(s_{\text{obj}}, a_{\text{base}})| \\ - \tau_{\text{time}}, \end{aligned} \tag{2}$$

where d_{obj} is the distance to the object; Δd_{obj} is the change in d_{obj} after performing a_{base} in the state s_{obj}; collision$(s_{\text{obj}}, a_{\text{base}})$ returns *True* if there was a collision with the table after taking a_{base} in s_{obj}; IK$(s_{\text{obj}}, a_{\text{base}})$ is the set of IK solutions to the object grasp poses after taking actions a_{base} in state s_{obj}; $|\cdot|$ is set cardinality; τ_{time} is a constant time penalty at each time step, and α_1, α_2 and α_3 are hyper-parameters.

Layer 2: In Layer 2, a binary action a_{grasp} 'grasp or not' is learned while using the policy π_{base} already learned in Layer 1 for taking the action a_{base}. Thus, the agent uses a composite policy for exploration, and only π_{grasp} has learnable parameters ϕ_{grasp} (see Fig. 2(b)):

$$a_{\text{grasp}} \sim \{\pi_{\text{base}}(\,\cdot\,|s_{\text{obj}}), \pi_{\text{grasp}}(\,\cdot\,|s_{\text{obj}}, a_{\text{base}};\ \phi_{\text{grasp}})\}\,. \tag{3}$$

This reduces the exploration space as only a_{grasp} needs to be explored. The extrinsic reward for learning policy π_{grasp} is calculated as:

$$r^{\text{e}}_{\text{grasp}}(s_{\text{obj}}, a_{\text{base}}, a_{\text{grasp}}) = \alpha_4 \cdot |\text{IK}(s_{\text{obj}}, a_{\text{base}}, a_{\text{arm}})|, \tag{4}$$

where IK$(s_{\text{obj}}, a_{\text{base}}, a_{\text{arm}})$ is the set of IK solutions to the object grasp candidates after taking actions a_{base} and a_{grasp} in state s_{obj}; $|\cdot|$ is set cardinality, and α_4 is a hyper-parameter. Further, π_{grasp} is modeled as a sequential policy by feeding the already learned action a_{base} in Layer 1 as an additional state. This allows the policy π_{grasp} to learn action a_{grasp} that when executed along with action a_{base} maximizes the reward $r^{\text{e}}_{\text{base}} \cup r^{\text{e}}_{\text{grasp}}$.

Layer 3: In Layer 3, the pre-grasp manipulator motion action a_{arm} is learned while using the policies π_{base} and π_{grasp} learned in Layers 1 & 2 for taking actions a_{base} and a_{grasp}. Thus, similar to Layer 2, the agent uses a composite policy for exploration, and only ϕ_{arm} has learnable parameters (see Fig. 2(c)):

$$\begin{aligned} a_{\text{arm}} \sim \{\pi_{\text{base}}(\,\cdot\,|s_{\text{obj}}), \pi_{\text{grasp}}(\,\cdot\,|s_{\text{obj}}, a_{\text{base}}), \\ \pi_{\text{arm}}(\,\cdot\,|s_{\text{obj}}, s_{\text{arm}}, a_{\text{base}}, a_{\text{grasp}};\ \phi_{\text{arm}})\}. \end{aligned} \tag{5}$$

The extrinsic reward $r^{\text{e}}_{\text{arm}}$ for learning policy π_{arm} is defined based on the IK solution which has a minimum difference to the robot's current shoulder and elbow joint angles (s_{arm}) as follows:

$$\begin{aligned} r^{\text{e}}_{\text{arm}}(\mathcal{S}, \mathcal{A}) = \mathbf{1}(a_{\text{grasp}}) \cdot \frac{\alpha_5}{1 + \min\limits_{e \in \text{IK}(\mathcal{S},\mathcal{A})} |e - s_{\text{arm}}|} \\ + \alpha_6 \cdot \mathbf{1}(\text{collision}(\mathcal{S}, \mathcal{A})), \end{aligned} \tag{6}$$

where α_5 and α_6 are hyper-parameters; collision$(\mathcal{S}, \mathcal{A})$ returns *True* if there was a self-collision or collision with the base platform and positive component of $r^{\text{e}}_{\text{arm}}$ is awarded only when a_{grasp} is *True*. Reducing the difference between shoulder & elbow joint angles and the selected IK solution for grasping directly contributes to the reduction in grasp execution time t_{grasp}. Further, π_{arm} is also modeled as a sequential policy by feeding already learned actions a_{base} and a_{grasp} as additional states. This allows the policy π_{arm} to learn action a_{arm} executed along with actions a_{base} & a_{grasp} learned in Layers 1 & 2 maximize the reward $r^{\text{e}}_{\text{base}} \cup r^{\text{e}}_{\text{grasp}} \cup r^{\text{e}}_{\text{arm}}$.

C. Robot Base Pose Prediction Network (BP-Net)

Learning suitable pre-grasp manipulator motions is a hard problem as the positive extrinsic reward $r^{\text{e}}_{\text{arm}}$ from the environment is only available when the robot attempts grasp. The IK solutions to reach the grasp poses depend on the base pose selected for grasping. In our problem setup, the base pose for

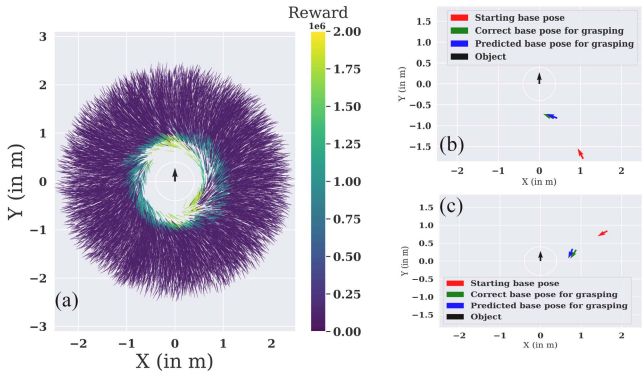

Fig. 3. (a) Data used for training the BP-Net. (b) and (c): Quality of base pose for grasping predictions made by the BP-Net.

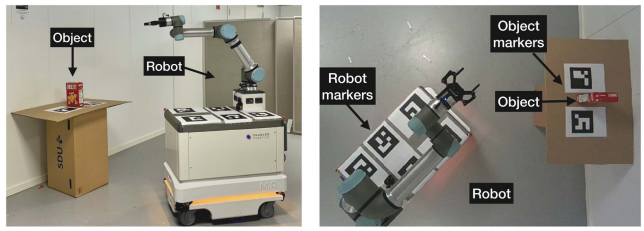

Fig. 4. Real-world experiment setup. (a) Object and robot platform. (b) State estimation using overhead camera.

grasping becomes known to the robot only in the terminal state of the RL episode when the policy π_{grasp} decides to attempt the grasp and end the episode. Hence, it is difficult to provide any dense rewards to guide the learning of the *pre-grasp manipulator motion* policy π_{arm} in Layer 3.

To address this problem, we introduce a *Robot Base Pose Prediction Network* (BP-Net). It predicts the base pose for grasping given the current object pose. The BP-Net is trained using the interactions from the previously learned *grasp decision* policy π_{grasp} in Layer 2. Given the object pose in the robot base frame s_{obj}, the network learns to predict $s_{\text{obj}}^{\text{grasp}}$. Since the mobile robot operates in 3D space, we reduce the dimensionality of the learning problem from 6D to 3D by converting the 6D object pose in a base frame s_{obj} to just a 3D robot base pose (x, y, θ) using robot's body transformations.

For predicting $s_{\text{obj}}^{\text{grasp}}$, we used a Multi-layer Perceptron with 5 hidden layers with 128, 512, 4096, 512, and 128 neurons respectively, and RELU activation function after the first 4 hidden layers. The network is trained with Mean-Squared Error (MSE) loss offline (outside the RL framework) with the data $(s_{\text{obj}}, s_{\text{obj}}^{\text{grasp}})$ collected using the policy π_{grasp} as shown in Fig. 3(a). The learning objective is to predict the robot base pose for grasping (base poses with high rewards in Fig. 3(a)) given any starting base pose (base pose with smaller rewards in Fig. 3(a)). We use $\hat{s}_{\text{obj}}^{\text{grasp}}$ to denote the predicted state by the network for grasping. The network trained is trained on 200,000 samples. Some qualitative base pose predictions are shown in Fig. 3(b) and (c). In Fig. 3, the black arrow indicates the object and the blue circle indicates a table.

We use this base pose prediction to provide shaped intermediate reward $r_{\text{arm}}^{\text{s}}$ to accelerate the learning of *pre-grasp manipulator motion* policy π_{arm}. Shaped reward $r_{\text{arm}}^{\text{s}}$ is provided before the robot has reached the base pose for grasping i.e. it is still approaching the target object. It is computed similar to the $r_{\text{arm}}^{\text{e}}$ in (6). However, IK solutions to the grasp poses are computed using the base pose for grasping $\hat{s}_{\text{obj}}^{\text{grasp}}$ predicted by BP-Net

$$r_{\text{arm}}^{\text{s}}(\mathcal{S}, \mathcal{A}) = \frac{\alpha_7}{1 + \min\limits_{e \in \text{IK}, (\hat{s}_{\text{obj}}^{\text{grasp}})} |e - s_{\text{arm}}|}. \tag{7}$$

where α_7 is a hyperparameter. This shaped reward $r_{\text{arm}}^{\text{s}}$ indicates relevant exploration spaces to the agent already at early stages of approaching for getting high $r_{\text{arm}}^{\text{e}}$.

V. Experimental Setup

A. Experiment and Implementation Details

For the evaluation of our proposed approach, we created an environment in NVIDIA Isaac Sim using the mobile manipulator platform and a table with the object to be grasped as shown in Fig. 1. Three grasp poses for this object were pre-defined; two from the sides and one from the top. Moreover, instead of relying on pre-defined grasp poses, grasp proposal networks such as [26] can also be used to provide rewards.

In each episode, the robot's starting pose was randomly sampled within a 2.5-3 m radius around the object and it could execute up to 25 steps using actions a_{base} and a_{arm} before the episode terminates. Executing an action a_{grasp} also terminates the episode. The state $\mathcal{S} = \{s_{\text{obj}}, s_{\text{arm}}\}$ was calculated based on the states provided by the simulator. For real-world deployment, we used the mobile manipulator platform as shown Fig. 4(a). The state s_{obj} was estimated with ArUco fiducial markers using an overhead camera as shown in Fig. 4(b).

For the algorithmic implementation of our approach, we used the Mushroom RL library [27]. The policies π_{base} and π_{arm} (that outputs continuous actions) were trained using Soft Actor-Critic (SAC) [28], while policy π_{grasp} (that outputs discrete action) was trained using Hybrid RL (HyRL) [8]. For all agents, an MLP with 3 hidden layers, each comprising 128 neurons with ReLU activation, was employed to train both the actor and critic. The learning rate was set to 1e-4. τ_{grasp} and τ_{time} were set to 0.8 and 1e4. Furthermore, α_1, α_2, α_3, α_4, α_5, α_6, and α_7 were set to 5e3, -1e5, 1e5, 1e5, 5e5, -5e4, and 5e4, respectively.

B. Experiment Objectives and Baselines

First, we compare the execution time and success rate of our method against representative baselines, which also involve explicit robot base placements for grasping (Section VI-A). We assume that the robot has access only to uncertain object pose estimates. The following baselines are considered:

Pre-defined base poses (PBP): The robot selects the base pose near the table that is closest to its starting pose as it doesn't have any information regarding the object pose.

Min. Distance & IK (MDI): The robot selects the base pose with a valid Inverse Kinematics (IK) solution using an Inverse Reachability Map (IRM) [9], [10]. If multiple solutions exist then the base pose closest to the robot's starting pose is selected.

Min. Navigation cost & IK (MNI): The robot selects the base pose with a valid IK solution that minimizes the navigation

time cost while maximizing the number of IK solutions to the grasp poses.

Min. Navigation cost & Max. Manipulability (MNMM): The robot selects the base pose and a pre-grasp pose that minimizes the navigation time cost and maximizes manipulability. The robot then simultaneously reaches this pre-grasp pose while navigating to the base pose. This is similar to [11] adapted for single object manipulation.

PBP, MDI, MNI, and MNMM represent conventional planning methods, where the plan is solely based on the robot's starting pose and the estimated object pose at that time. As planning is such a high-dimensional space is computationally expensive, in case of potential failures, these methods were allowed to re-plan up to 3 times using updated robot and object poses.

Second, we analyze the learning performance of the proposed approach (Section VI-B). For this analysis, we compare our approach against the following learning methods:

Hybrid-RL (HRL): The agent learns all the actions together. Since the robot's action space consists of continuous ($a_{\text{base}}, a_{\text{arm}}$) and discrete actions ($a_{\text{grasp}}$), Hybrid RL [8] is used for learning. The action and state space are the same as in 'Ours', and the maximum reward the agent can obtain at any time step is $r^{\text{e}}_{\text{base}} \cup r^{\text{e}}_{\text{arm}} \cup r^{\text{e}}_{\text{grasp}}$. To ensure a fair comparison, we provide HRL with the same number of experiences as the total experiences used for training our agent across the three layers.

Curriculum Learning (CL): Similar to our approach, CL also learns in stages using a curriculum [16]. However, only a single policy is used for learning all the actions unlike 'Ours' where each individual action is learned using a separate policy. At each learning stage, a new action is introduced, and a new policy is learned from scratch for all the actions. The Q-function from the policy learned in the previous learning stage is used to guide exploration [29]. Additionally, the new extrinsic reward is introduced at each subsequent learning stage, as it is directly connected to the newly introduced actions. However, unlike 'Ours', the CL does not have access to the shaped reward $r^{\text{s}}_{\text{arm}}$.

Third, we compare learning performance with and without shaped reward in Section VI-C.

Finally, we show the performance of our method and representative baselines while using the ground truth and noisy pose estimates without re-planning (Section VI-D).

VI. Results

A. Comparison With Mobile Manipulation Methods

In Table I, we present the mean and standard deviation values for navigation and manipulation execution time, total execution time, and success rate over 2000 runs for the four chosen baselines and our proposed method ('Ours'). During manipulation, the maximum joint velocity and acceleration limits for the UR5e manipulator were set to 0.5 rd/s and 0.25 rd/s^2, respectively, while during navigation, the maximum base linear and angular velocity was set to 0.1 m/s and 0.25 rd/s. To make the task more realistic, we introduced random Gaussian noise proportional to 1% of the object-to-robot distance (greater distance, larger noise).

TABLE I
Comparison of Execution Time & Success Rate

Method	Navigation time (in secs)	Manipulation time (in secs)	Total time (in secs)	Success rate (in %)
PBP	22.86 ± 10.42	11.09 ± 1.28	33.95	72.48
MDI	17.91 ± 2.98	11.16 ± 1.21	29.07	95.48
MNI	25.77 ± 8.33	11.48 ± 0.93	37.25	94.74
MNMM	25.86 ± 8.66	10.29 ± 2.82	36.15	90.07
Ours	**13.05** ± 3.93	**8.30** ± 0.21	**21.35**	**96.02**

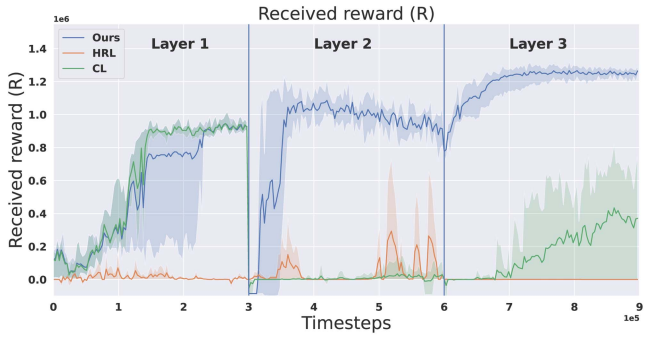

Fig. 5. Comparing the learning performance of our work (Ours) with methods HRL and CL.

As shown in Table I, our method outperforms the baselines in terms of time performance. Failures for all methods can occur either because the base pose would result in a collision with the workspace or due to the unavailability of IK solutions for the grasp poses at the planned base pose for grasping. Baselines re-plan when potential failures are detected, leading to increased execution times. In contrast, our method continuously incorporates updated object poses during action prediction, resulting in improved performance.

B. Comparison With Learning-Based Approaches

In Fig. 5, we present a comparison of the learning performances of the different methods. For each approach, we calculate the mean over five seed runs and also report the minimum and maximum variations. The vertical lines at $t = 300$K and $t = 600$K indicate the transitions from Layer 1 to Layer 2 and from Layer 2 to Layer 3 learning, respectively. In the case of method CL, the vertical lines indicate the introduction of new actions. For method HRL, vertical lines have no meaning. As mentioned in Section VI-B, the maximum rewards that the baselines and 'Ours' can obtain at any time are not the same. However, at Stage 3, all three methods have access to almost the same maximum reward (except that our method also has access to $r^{\text{s}}_{\text{arm}}$, but its contribution is negligible compared to other rewards and hence can be ignored). From the results, it is evident that our proposed method clearly outperforms both the CL and HRL approaches.

The HRL method which tries to learn all three actions together fails to learn anything even after being trained for 900 K time steps. This is because while jointly exploring all three actions, it fails to learn the approaching base motion for grasping. And without learning the base motion it also fails to learn the pre-grasp manipulator motion and to determine when to grasp as

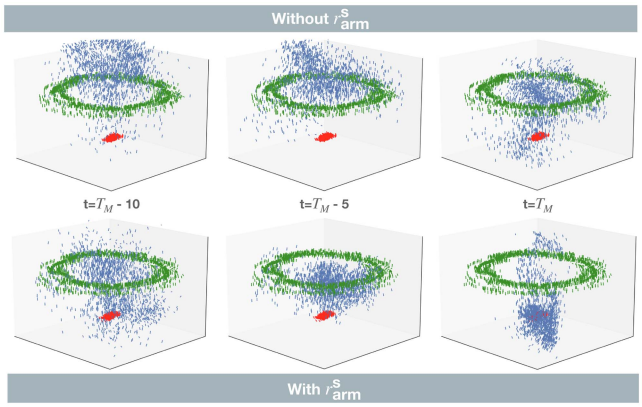

Fig. 6. End effector poses at different time steps during the approaching motion for 2000 different samples. Red: Targeted grasp poses Green: End effector pose at grasping without any pre-grasp manipulator motion Blue: End-effector pose at grasping after pre-grasp manipulator motion.

both of these actions have sparse rewards that cannot be obtained without having learned how to approach the object.

The CL method uses a curriculum but tries to learn all the actions using a single policy. During the first 300 K steps, it learns a_{base}, then during the next 300 K steps, it learns a_{base} & a_{grasp} and then learns all three actions during the final 300 K steps. It successfully learns to approach the object for grasping with base motion as can be seen in Fig. 5. However, it fails to learn a good policy when it tries to learn a_{base} and a_{grasp} together using the Q-function from the previously learned policy for action a_{base}. This is because a_{grasp} is an episode-ending action and prevents exploration for learning a_{base}. Finally, the robot completely refrains from attempting to grasp as it ends the episode and prevents it from obtaining the rewards from other actions. It ends up learning an undesired policy wherein it tries to slow down its base velocities after reaching within the grasping distance of the object but never attempts to grasp.

As a result, both the HRL and CL methods achieved a 0% success rate and hence were not included in Table I.

C. Impact of the Shaped Reward r^s_{arm}

In this study, we show the difference between the quality of learned pre-grasp manipulator motions with and without shaped reward $r^{\text{s}}_{\text{arm}}$. In Fig. 6, we plot end-effector poses at different time steps during the approaching motion. T_M is the time when the robot attempts the grasp. Thus end-effector poses at $t = T_M$ are the actual end-effector poses when the robot attempted to grasp. End-effector poses at $t = T_M - x$ indicate how the end-effector poses would have looked like at $t = T_M$ if the robot had stopped moving the arm at $t = T_M - x$. The first and second row in Fig. 6 describe end-effector poses during pre-grasp manipulator motion without and with $r^{\text{s}}_{\text{arm}}$ respectively. It can be seen that at $t = T_M$ using $r^{\text{s}}_{\text{arm}}$ results in better end-effector poses (blue) which are much closer to the targeted grasp poses (red). It can be also seen that at $t = T_M - 5$, the policy learned with $r^{\text{i}}_{\text{arm}}$ already have good end-effector poses that are close to object grasp candidates.

D. Impact of Noise Without Re-Planning for Baselines

In Fig. 7, we show the impact of noise by comparing the time performance and success rate while using the ground truth and uncertain object pose estimates without re-planning for the baseline methods. PBP does not use the object pose estimates, so it is not affected by the noise. In contrast, all the other methods, including 'Ours,' attain a 100% success rate when the ground truth object pose estimates are used. In case of no noise in the object pose estimate, MNMM outperforms our method as it generates a plan with similar objectives as 'Ours' upfront and executes it without considering any new information about the object's pose. However, in the presence of uncertainty in the object pose estimate, it only achieves around 20% success rate.

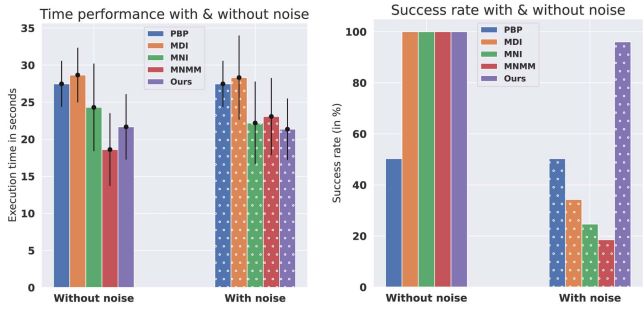

Fig. 7. Time performance & success rate without re-planning while using ground truth & noisy object pose estimates.

E. Real-World Experiments

Learned policies can be transferred to real-world environments, as shown in the supplementary video. For quantitative evaluation, we conducted 20 trials, with no failures due to collisions with the workspace or unavailability of IK solutions at the selected base pose for grasping. Failures resulting from poor pose estimates during the final grasp were disregarded, and the trial was repeated. Our approach achieved a total execution time (approaching + grasping) of $41.66\,\text{s} \pm 5.54$ seconds. For comparison, we measured the total execution time if the robot had stayed in the default manipulator configuration, resulting in 58.28 ± 5.83 seconds. Thus, adopting a suitable pre-grasp manipulator configuration reduced execution time by almost 28.51%.

VII. Conclusion and Future Work

In this work, we have presented a layered learning approach for learning suitable pre-grasp manipulator configurations while navigating to the target object for grasping to optimize the time cost of mobile manipulation. We compared our work with four baselines that use conventional planning and show that in the presence of uncertain object pose estimates, the proposed approach results in reduced execution time. We also show that the learned policy can be successfully deployed on a real robot.

The policies learned with the object placed at the center of the circular table can be applied to any convex workspace, regardless of the object's placement on the table using a recovery policy [30]. The recovery policy can be a simple controller with obstacle avoidance and workspace-following capabilities to direct the robot to safe zones when the actions predicted by the policy face execution challenges. Moreover, policies can be trained to generalize across diverse object poses by representing the state in the workspace frame and randomly sampling various object poses during training. This, however, requires a significantly larger amount of data.

Future work can involve the integration of navigation costmaps during the approaching base motion. Further, since our approach only uses shoulder and elbow joints for the pre-grasp manipulator motion, the remaining joints can be used, for example, for planning the robot's "eye in hand" camera views for improving object pose estimation.

ACKNOWLEDGMENT

The authors would like to thank the SDU eScience Center for providing computational resources on the UCloud interactive HPC system and the SDU I4.0 lab for providing the robot for experimental evaluation.

REFERENCES

[1] A. Dömel, S. Kriegel, M. Kaßecker, M. Brucker, T. Bodenmüller, and M. Suppa, "Toward fully autonomous mobile manipulation for industrial environments," *Int. J. Adv. Robot. Syst.*, vol. 14, no. 4, 2017, Art. no. 1729881417718588.

[2] J. Carius, M. Wermelinger, B. Rajasekaran, K. Holtmann, and M. Hutter, "Deployment of an autonomous mobile manipulator at MBZIRC," *J. Field Robot.*, vol. 35, no. 8, pp. 1342–1357, 2018.

[3] D. Honerkamp, T. Welschehold, and A. Valada, "N^2 M^2: Learning navigation for arbitrary mobile manipulation motions in unseen and dynamic environments," *IEEE Trans. Robot.*, vol. 39, no. 5, pp. 3601–3619, Oct. 2023.

[4] J. Haviland, N. Sünderhauf, and P. Corke, "A holistic approach to reactive mobile manipulation," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 3122–3129, Apr. 2022.

[5] D. Honerkamp, T. Welschehold, and A. Valada, "Learning kinematic feasibility for mobile manipulation through deep reinforcement learning," *IEEE Robot. Automat. Lett.*, vol. 6, no. 4, pp. 6289–6296, Oct. 2021.

[6] C. Wang et al., "Learning mobile manipulation through deep reinforcement learning," *Sensors*, vol. 20, no. 3, 2020, Art. no. 939.

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

[8] S. Jauhri, J. Peters, and G. Chalvatzaki, "Robot learning of mobile manipulation with reachability behavior priors," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 8399–8406, Jul. 2022.

[9] A. Makhal and A. K. Goins, "Reuleaux: Robot base placement by reachability analysis," in *Proc. IEEE Int. Conf. Robot. Comput.*, 2018, pp. 137–142.

[10] N. Vahrenkamp, T. Asfour, and R. Dillmann, "Robot placement based on reachability inversion," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2013, pp. 1970–1975.

[11] F. Reister, M. Grotz, and T. Asfour, "Combining navigation and manipulation costs for time-efficient robot placement in mobile manipulation tasks," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 9913–9920, Oct. 2022.

[12] S. Vafadar, A. Olabi, and M. S. Panahi, "Optimal motion planning of mobile manipulators with minimum number of platform movements," in *Proc. IEEE Int. Conf. Ind. Technol.*, 2018, pp. 262–267.

[13] A. Dietrich, T. Wimbock, A. Albu-Schaffer, and G. Hirzinger, "Reactive whole-body control: Dynamic mobile manipulation using a large number of actuated degrees of freedom," *IEEE Robot. Autom. Mag.*, vol. 19, no. 2, pp. 20–33, Jun. 2012.

[14] L. Naik, T. M. Iversen, A. Kramberger, J. Wilm, and N. Krüger, "Multi-view object pose distribution tracking for pre-grasp planning on mobile robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 1554–1561.

[15] F. Stulp, A. Fedrizzi, L. Mösenlechner, and M. Beetz, "Learning and reasoning with action-related places for robust mobile manipulation," *J. Artif. Intell. Res.*, vol. 43, pp. 1–42, 2012.

[16] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," *J. Mach. Learn. Res.*, vol. 21, pp. 7382–7431, 2020.

[17] P. Stone and M. Veloso, "Layered learning," in *Proc. Eur. Conf. Mach. Learn.*, 2000, pp. 369–381.

[18] Y. Meng, Y. Chen, and Y. Lou, "Uncertainty aware mobile manipulator platform pose planning based on capability map," in *Proc. IEEE Int. Conf. Real-Time Comput. Robot.*, 2021, pp. 123–128.

[19] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. 26th Int. Conf. Mach. Learn.*, 2009, pp. 41–48.

[20] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[21] J. Eschmann, "Reward function design in reinforcement learning," in *Reinforcement Learning Algorithms: Analysis and Applications*. Cham, Switzerland: Springer, 2021, pp. 25–33.

[22] B. F. Skinner, *Science and Human Behavior*. New York, NY, USA: Simon and Schuster, 1965.

[23] R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, and D. Pathak, "Planning to explore via self-supervised world models," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 8583–8592.

[24] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," 2018, *arXiv:1810.12894*.

[25] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2778–2787.

[26] J. Mahler et al., "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *Proc. Robot.: Sci. Syst. XIII*, 2017.

[27] C. D'Eramo, D. Tateo, A. Bonarini, M. Restelli, and J. Peters, "MushroomRL: Simplifying reinforcement learning research," *J. Mach. Learn. Res.*, vol. 22, no. 131, pp. 5867–5871, 2021.

[28] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actorx," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.

[29] G. Farquhar, L. Gustafson, Z. Lin, S. Whiteson, N. Usunier, and G. Synnaeve, "Growing action spaces," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 3040–3051.

[30] B. Thananjeyan et al., "Recovery RL: Safe reinforcement learning with learned recovery zones," *IEEE Robot. Automat. Lett.*, vol. 6, no. 3, pp. 4915–4922, Jul. 2021.