Decoupling skill learning from robotic control for generalizable manipulation

Kai Lu[1], Bo Yang[2], Bing Wang[1], Andrew Markham[1]

***Abstract*— Recent work in robotic manipulation through reinforcement learning (RL) or imitation learning (IL) has shown potential for tackling a range of tasks e.g. opening a drawer or a cupboard. However, these techniques generalize poorly to unseen configurations. We conjecture that this is due to the high-dimensional state space for joint control. In this paper, we take an alternative approach and separate the task of learning 'what to do' from 'how to do it' i.e. whole-body control. We pose the reinforcement learning problem as one of determining the skill dynamics for a disembodied virtual manipulator interacting with articulated objects. The whole-body robotic kinematic control is optimized to execute the high-dimensional joint motion to reach the goals in the workspace. It does so by solving a quadratic programming (QP) model with robotic singularity and kinematic constraints. Extensive experiments on manipulating complex articulated objects show that our approach is generalizable to unseen objects with large intra-class variations, outperforming previous approaches. The benchmark evaluation indicates that our approach produces more compliant robotic motion and outperforms the pure RL and IL baselines in task success rates.**

I. INTRODUCTION

Robotic manipulation has a broad range of applications, such as industrial automation, healthcare, and domestic assistance. For repetitive tasks in controlled environments, e.g. automotive assembly, robotic manipulation has enjoyed many years of success. However, commonly used methods, such as model-predictive control and off-the-shelf planners, typically require accurate physical models of objects, and the environment, which are largely unavailable in uncontrolled settings 'in-the-wild'. Learning-based methods have recently been studied in household manipulation tasks [1]–[4], where a visual control policy is learned from interactions via reinforcement learning (RL), or learned from demonstrations via imitation learning (IL). However, a common concern of RL is the unproductive exploration in the high-dimensional continuous action space and state space of a whole-body robot. IL, on the other hand, usually suffers from the distribution shift problem due to the lack of high-quality demonstrations over numerous scenarios [5]. Hence, it remains very challenging for high-DOF robots to learn complex manipulation skills and further generalize the skills to novel objects.

We note that humans and other animals can easily generalize a learned skill from one object to another, and execute the skills even if a limb is injured or being constrained.

[1]: K. Lu, B. Wang, and A. Markham are with the Departments of Computer Science, University of Oxford, Oxford, UK. {kai.lu, bing.wang, andrew.markham}@cs.ox.ac.uk

[2]: B. Yang is with the Department of Computing, Hong Kong Polytechnic University, Hong Kong, China. bo.yang@polyu.edu.hk

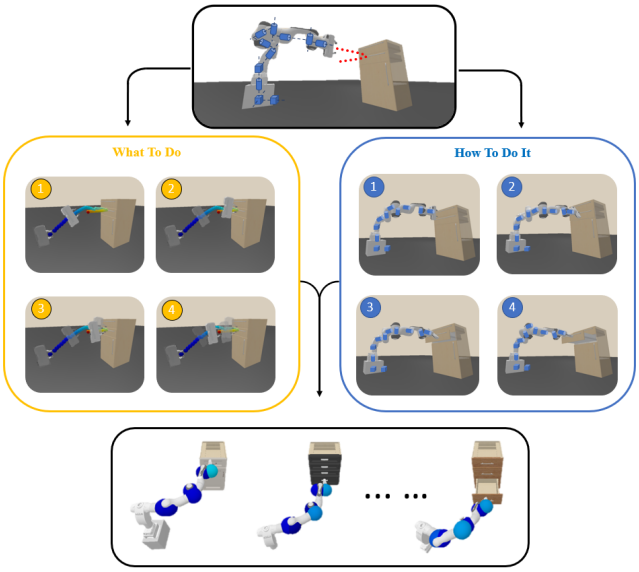


Fig. 1. **Demonstration of decoupling skill dynamics.** The left panel shows how we use reinforcement learning to determine how to control a disembodied or floating end-effector ('what to do'). The right panel shows how we use quadratic programming to realize a compliant trajectory and set of joint-torque control signals ('how to do it'). The bottom panel shows how we can use this to generalize to different, previously unseen scenarios.

We take a preliminary step towards providing robots with similar capabilities. To achieve this, we propose that the complex yet low-DOF *skill dynamics* (i.e., what should be done) can be decoupled from robot-specific kinematic control (i.e., how this behavior needs to be implemented). The skill dynamics themselves can be learned from manipulator-object interactions, without being concerned with high-DOF whole-body control. Whole-body control can be treated as a classical inverse kinematics problem. We conjecture that the robot could better learn generalizable manipulation skills through this decoupling, by making the RL tasks simpler.

The most similar works to ours are action-primitive methods. Dalal et al. [6] manually specify a library of robot action primitives such as 'lift' and then learn an RL policy to parameterize these primitives such as 'lift to position (x,y)'. Alternatively, Pertsch et al. [7] propose to learn a prior over basic skills from offline agent experience, which allows the robot to explore these skills with unequal probabilities. These methods decompose the agent action space into high-level policy and low-level predefined behaviors, hence substantially improve the learning efficiency. However, not all the manipulation skills can be represented as a series of action

Manuscript 2965 submitted to 2023 IEEE International Conference on Robotics and Automation (ICRA). Received September 15, 2022.

primitives. In this paper, we do not expect that a robot needs to learn skills by manually creating action libraries which might cause a time-consuming selection and recalibration when transferring skills to novel objects or tasks.

To realize our approach, We learn the skill dynamics with a 'disembodied' manipulator (effectively a free-floating end-effector) via model-free RL and then control the actual robot via QP optimization based on its full physical model. The general idea is appealing since learning skill dynamics from interactions enhances the robot's understanding of its surrounding scenes and objects, leading to diverse manipulation trajectories and generalizability to differing objects. With regards to the RL agent, controlling a disembodied manipulator greatly reduces the dimensionality and complexity of the search space, leveraging the strengths of data-driven methods. In our approach, the whole-body controller formulates a QP model with robotic singularity and kinematic constraints. The high-dimensional joint-space actions are produced by: 1) predicting end-effector (EE) poses and velocities from learned skill dynamics (i.e., the RL agent), 2) optimizing the joint velocities by a QP solver, 3) automatically generating joint torques by inner PID controllers of the simulator. Our method thus accelerates RL process and produces more compliant, smoother and controllable robot motions. In converse, using pure RL for controlling whole robots can generate jerky or 'locked' motions due to singularities, leading to a high task failure rate.

We evaluate this approach using the ManiSkill [1] robotic simulated environment consisting of a 13-DOF Franka robot [8], onboard RGB-D cameras providing point cloud observations, and a variety of articulated objects. Experiments show that our approach can learn generalizable skills over different cabinets of the training sets and unseen test sets. We achieve an average success rate of 74% in training cabinets and a 51% in test cabinets in the drawer opening task, significantly outperforming existing techniques. We show that the generalization of our model is improved by increasing the number of training scenes. We also compare the robotic motions produced by our method and pure RL, showing that robot singularities are avoided by our QP optimization.

II. RELATED WORK

Robot learning for manipulation skills has been intensively studied in recent years. The dominant methods are reinforcement learning and imitation learning. An extensive survey of existing methods can be found in Kroemer's work [10]. However, robotic manipulation of complex objects through controlling high-DOF robots remains a challenging problem.

Recent works apply a variety of RL algorithms for robotic manipulation [11]–[14]. However, productive exploration has always been a challenge for robotic RL due to complex dynamics of high-DOF robots. Many works have look into decomposing the action space into higher-level policy and lower-level control to improve the sampling efficiency and the task performance. Exploiting predefined action primitives (e.g., lift, move, slide) is a convenient way to reduce the action space dimensions [6], [7], [15], but the it requires

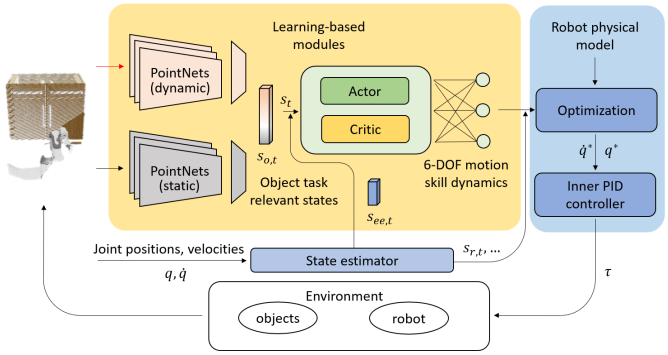


Fig. 2. **Architecture of the proposed system.** As shown in the yellow block, we use two simple PointNet [9] modules to separately perceive static (e.g. size of object) and dynamic (e.g. current position of handle) states. These are input to a SAC RL framework to learn how to control the disembodied end-effector, realizing a 6-DOF motion skill. Through knowledge of the robot physical model, QP is used to effect optimal control of the joint dynamics of the whole-body robot.

manual specification of the robot behaviors. Another line of research regarding choosing action space looks into operational space control (OSC) [16]–[18], which requires a precise mass matrix to map motor commands by EE motions into joint torques. Martin et al. [19] compare OSC and other control modes in EE space and joint space using Robosuite [20] simulated environment and demonstrate that OSC usually performs better than other controllers in contact-rich tasks. Although the motivations of our method and OSC-based RL are similar, our agent is a disembodied manipulator instead of a whole-body robot, which does not require high-fidelity modelling of mass matrix as OSC does. In the presence of inaccuracies in dynamics modeling, OSC's performance quickly deteriorates [21]. Furthermore, this work presents the generalizability over a variaty of novel objects in ManiSkill environment, while tasks in Robosuite typically focus on a single scene.

Imitation learning is usually considered as a more sampling-efficient approach, but high-quality demonstrations that can be directly employed to robot motors are usually difficult to collect [5]. Shen et al. [22] propose a generative adversarial self-imitation learning method and a balancing expert buffer to mitigate the issues in ManiSkill's demonstration trajectories.

Another highly related topic is robotic vision for articulated object manipulation. Mo et al. [23], Wu et al. [24] and introduce encoder-decoder networks to extract affordance maps and predict action proposals or trajectory proposals attached to the objects 3D representations. Eisner et al. [25] also implement a neural network to estimate point-wise motion directions that they call articulation flow for the point cloud observations of objects. Similarly, Xu et al. [26] propose to estimate the motion direction and potential moving distance of the articulated part but only using a single image as input. In contrast, Mittal et al. [27] explicitly infer the articulation properties such as joint state and part-wise segmentation by given the object category. These methods are object-centric representation learning and more focus

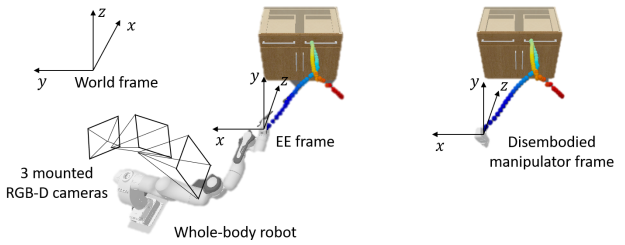


Fig. 3. **Coordinate system of our method.** During the manipulation process, we optimize the robot joint actions to approximate the EE to the disembodied manipulator's trajectory. At every time step, the point cloud observation is obtained from the three cameras mounted on the robot.

on object articulation properties, usually leaving the next manipulation stage in a simplified way (e.g., a perfect suction cup in [26]), while our work learn the agent-centric policy of a disembodied hand whose actions are continuous and close-loop in the simulated environment.

III. Reinforcement Learning for Skill Dynamics

Fig. 2 shows the overall framework of our approach. We first learn the control policy of a disembodied manipulator as shown in the yellow block, and then compute the whole robot's high-dimensional joint actions by QP optimization as illustrated in the light blue block. We discuss the RL policy and visual perception in the current section, leaving the discussion of the whole-body controller in section IV.

A. Problem Formulation

Robotic manipulation skill learning is usually formulated as a Markov decision process (MDP), which is represented as (S,A,R,T,γ) , where $S \in \mathbb{R}^n$ is the set of states, $A \in \mathbb{R}^m$ is the set of actions, $R(s_t,a_t,s_{t+1})$ is the reward function, $T(s_{t+1}|s_t,a_t)$ is the transition function as a probability distribution, and γ is the discount factor for the future rewards. The agent policy $\pi(a|s)$ represent the action selecting probability under a given state s. The goal of RL is to maximize the return under the policy $G_\pi = \mathbb{E}_\pi[\sum_t \gamma^t R(s_t,a_t,s_{t+1})]$. In vision-based RL, we usually need to estimate the task-relevant states from observation O. This setting is viewed as the partially observable Markov decision process (POMDP) and the policy is regarded as $\pi(a|f(o))$ where $f(o)$ is the estimated states from visual representations.

B. Disembodied Manipulator Environment

The agent for learning skill dynamics is a 6-DOF disembodied (or floating/ flying) manipulator with two fingers (as shown in Fig. 3). The DOF of manipulator is realized by three virtual prismatic joints and three virtual revolute joints. Therefore, the action space is six desired velocities of the virtual joints and two desired positions of the finger joints. Following the settings in ManiSkill, the action space is normalized to $(-1,1)$ in RL training.

$$a = [\dot{q}_0^*, \dot{q}_1^*, \dot{q}_2^*, \dot{q}_3^*, \dot{q}_4^*, \dot{q}_5^*, q_6^*, q_7^*,] \tag{1}$$

The state space of a manipulation task contains robot states s_r and target object states s_o:

$$s = [s_o, s_r] \tag{2}$$

For the cabinet environment of ManiSkill, we define $s_o \in \mathbb{R}^{m_1}$ as the cabinet's pose in world frame coordinates (x,y,z), the quaternion for rotation, and the pose of the goal link, joint, and handle, $s_r \in \mathbb{R}^{m_2}$ is the disembodied manipulator's or the whole robot's joint positions and velocities.

We follow the original design of dense reward function in the ManiSkill environment which is verified by the trajectories found by MPC-CEM [1]. For the cabinet opening task, the dense reward measures the distance between the manipulator and the handle of the target link (i.e., drawer or door), encourages the correct motion of the target link, and expect the scene to be static after finishing the task.

$$R = \begin{cases} R_{stage1} + R_{ee}, \ d > 0.01 \\ R_{stage2} + R_{ee} + R_{obj}, \ d < 0.01 \ and \ k_q < 0.9 \\ R_{stage3} + R_{ee} + R_{obj} + R_{static}, \ d < 0.01 \ and \ k_q > 0.9 \end{cases} \tag{3}$$

where the distance between EE and the target drawer handle d and opening extent of the the target drawer's joint $k_q = \frac{q_{cur} - q_{min}}{q_{max} - q_{min}}$ are used to define the stages.

C. Soft Actor-Critic Algorithm for Manipulation

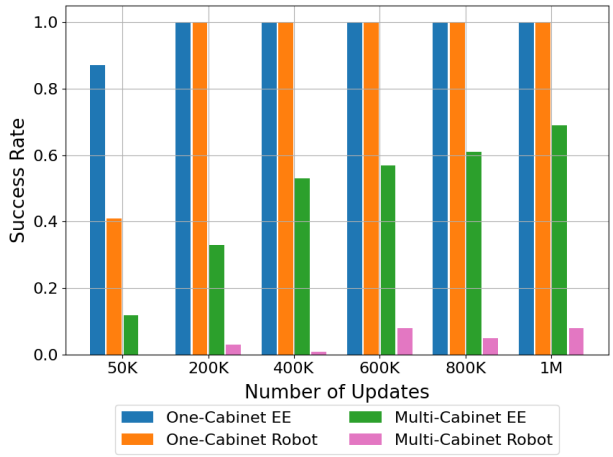
To learn the skill dynamics, we implement a model-free soft actor-critic (SAC) [28] algorithm for policy learning. We use the Gaussian policy head represented as $\pi(a|s)$ for our continuous action space. The actor and critic functions are approximated by multi-layer perceptions (MLP). The general model is computationally efficient since we only use 5-layer MLPs with a maximum size of 512. The parameters of the MLPs are updated as off-policy RL with the samples from replay buffer $D_{replay} = \{(s_t,a_t,r_t,s_{t+1})|t = 0,1,...,k\}$.

The learning curves of our RL agent of the disembodied manipulator and the original SAC agent of whole robot by ManiSkill-Learn [1] are shown in Fig. 4a. For an identical cabinet, both the robot and the manipulator successfully learn the specific skill, but the manipulator learns more quickly. We notice that the whole body robot finally obtains a higher episode average reward than the manipulator because its EE can move much faster when it does not fall into robotic singularities and hence completes the task in a shorter episode length. However, the results on testing across multiple different cabinets show that the whole robot configuration rarely succeeds to learn the required manipulation skill due to its high-dimensional action space, whilst our manipulator can still learn skill dynamics over a variety of objects with excellent success rates.

D. 3D Visual Perception

The vision module of our work is regarded as a state estimator, mapping 3D observations to states s_o. Similar to the benchmark baselines [1], we use PointNet modules [9] as the backbone which takes masked point clouds as input. To achieve a more stable state estimation, we use two separate

(a) Learning curves of RL models.



(b) Evaluation on training cabinets.

Fig. 4. **Comparison of the disembodied manipulator agent and the whole-body robot agent.** We compare the RL agents under two conditions: training with one cabinet or multiple (e.g., 15) different cabinets. Although they both succeed in one-cabinet environment, disembodied manipulator performs much better in multiple-cabinet environment.

PointNets for estimating static states (e.g., the length of drawers) and dynamic states (e.g., the current pose of the handle) respectively. In this work, we use single-frame prediction and fully-supervised learning for the vision module. We leave unsupervised visual representation learning from interaction (LfI) [29]–[31] for future exploration.

The pairs of point clouds and states for training are collected by synchronizing the robot's EE and the disembodied manipulators via inverse kinematics (IK). Note that point clouds are partial observations of the cabinet since we only use three cameras mounted in the robot and one world camera fixed in front of the cabinet. We also synchronize the cabinets' states in robot's and manipulator's environments and then record observations from robot. The training set can be represented as $D_{vis} = \{(o_{pcd,t}, s_t) | t = 0, ..., k\}$. Finally, the PointNets followed by MLPs $\Phi(\cdot)$ are supervised with an L2 loss between ground truth $s_{o,t}$ and predicted $\hat{s_{o,t}}$.

$$\mathscr{L}_{vision} = ||s_{o,t} - \hat{s_{o,t}}||_2, \ \ where\ \hat{s_{o,t}} = \Phi(o_{pcd,t}) \tag{4}$$

Table I compares the performances of RL models with ground truth states and with visual estimated states. The task success rates of vision-RL are slightly decreasing due to visual state estimation errors. We observe that these errors lead to position shifts of the manipulator but the RL agent has sufficient robustness to these shifts in the opening drawer task. We believe the accuracy and success rates of manipulation can be improved by using better vision models.

IV. QP-Based Whole-Body Controller

The second phase of our framework is to calculate the joint space actions based on the robot physical model for whole-body control. Our aim is to control the robot's EE to execute the learned skill dynamics in the same way as the disembodied manipulator. The poses and velocities in the world frame of the manipulator are accurately calculated via forward kinematics (FK), and then we synchronize the robot EE to approximate the trajectory which can be viewed as an inverse kinematics (IK) problem. However, the robot in our environment consists of a movable base and the off-the-shelf IK solver usually returns solutions such that the EE displacement is mainly contributed by the base joints. Moving the base instead of the arm usually is slower and more energy consuming. To decrease the base motion while avoiding robotic singularities, we introduce QP optimization to calculate the joint velocities for controlling the EE's motion. In this work, we use Klampt [32] for robotic kinematics calculations and qpOASES [33] as the QP solver.

According to forward kinematics, the robot's EE pose $x \in \mathbb{R}^6$ are represented by the robot's configuration (i.e., joint positions or joint angles) q as,

$$x = p(q), q \in \mathbb{R}^n \tag{5}$$

The Jacobian matrix of EE in the current configuration indicates the influences of each joint's motion to EE, which can be written as,

$$J(q) = \begin{bmatrix} \partial p_0/\partial q_0 & \partial p_0/\partial q_1 & \dots & \partial p_0/\partial q_{n-1} \\ \partial p_1/\partial q_0 & \partial p_1/\partial q_1 & \dots & \partial p_1/\partial q_n \\ \vdots & \vdots & \ddots & \vdots \\ \partial p_m/\partial q_0 & \partial p_m/\partial q_1 & \dots & \partial p_m/\partial q_n \end{bmatrix} \tag{6}$$

where m is the DOF of the robot's workspace and n is the DOF of joint space.

$$\dot{x}_{ee} = \begin{bmatrix} \omega \\ v \end{bmatrix} = J(q)\dot{q} \tag{7}$$

The task of approximating the EE motion to the disembodied manipulator can be written as,

$$T_{ee} = J(q)\dot{q} - \dot{x}^*_{ee} \tag{8}$$

where $\dot{x}^*_{ee}$ is the desired pose change of robot EE. At timestamp t, we use the trajectory of the disembodied manipulator $\{x_{m,0}, x_{m,1}, ..., x_{m,k}\}$, to obtain $\dot{x}^*_{ee,t} = \partial(x_{ee})/\partial t = (x_{m,t+\triangle t} - x_{ee,t})/\triangle t$. The reason for not directly using the predicted actions $a_{m,t}$ by the RL policy as $x^*_{ee,t}$ is that these actions are the target joint velocities applied to manipulator's virtual joints so they do not indicate the real pose changes.

Robotic singularities refer to degenerate joint configurations that cause the rank of the Jacobian matrix to reduce. In the case of EE, if $rank(J_{ee})$ drops below the workspace dimension, then at q the robot cannot move instantaneously in a certain workspace dimensions. For example, in Eq. 6 $m = 6$ and the i-th line is equal to zero, the robot is in a singularity condition. It is dangerous when robots are at singularities or near-singular configurations, as these usually cause extreme accelerations of joints. To avoid singularities, we expect that the robot should be able to move the EE flexibly at the current configuration q. We attempt to parameterize this condition as the robot's capability of moving the EE to the desired pose after T_{sg} steps while the base is fixed. Therefore, we design the parameter of QP's cost function,

$$C_{sg} = [c_{base} I_{base}, c_{arm} I_{arm}, c_{finger} I_{finger}] \in \mathbb{R}^{n \times n} \tag{9}$$

where c is the weight parameter and I the identity matrix. We use the learned RL policy as a planner to predict the desired EE's pose after T_{sg} steps as $x^*_{ee,t+T_{sg}}$ and then perform IK,

$$z_{res}, q_{ik} = f_{ik}(q_t, x^*_{ee,t+T_{sg}}) \tag{10}$$

where $z_{res} = 1$ when IK is solved and $z_{res} = 0$ when failing to find a feasible configuration, which indicates that EE is unable to reach $x^*_{ee,t+T_{sg}}$ if the base remains static during t to $t + T_{sg}$. Then

$$\begin{aligned} c_{base} &= \omega_1 z_{res} + \omega_2 (1 - z_{res}) \\ c_{arm} &= \omega_2 z_{res} + \omega_1 (1 - z_{res}) \end{aligned} \tag{11}$$

QP is then applied to optimize desired joint velocities $\dot{q}_d$,

$$\begin{aligned} &\underset{\dot{q}_d}{\text{minimize}} \quad \dot{q}_d^T C_{sg} \dot{q}_d \\ &\text{subject to} \\ &\qquad J(q)\dot{q}_d = \dot{x}^*_{ee} \\ &\qquad \dot{q}_{\min} \le \dot{q}_d \le \dot{q}_{\max} \end{aligned} \tag{12}$$

where $\omega_1 > \omega_2$, the QP tends to encourage base motion when the robot's EE cannot move to $x^*_{ee,t+T_{sg}}$ with its base fixed, which accordingly avoids robotic singularities in most cases. In our experiment, we set $T_{sg} = 10, \omega_1 = 20, \omega_2 = 0.1$ for QP.

V. Experiments

In this section, we evaluate our approach by analyzing the model improvements during interactions, comparing the generalizability to novel objects in the benchmark, and estimating the motion compliance of the robot.

The experiments are performed in the ManiSkill simulated environment built on SAPIEN [34]. In particular, we apply our approach to the cabinet drawer opening task, where the success condition is opening the target drawer's joint to 90% of its extent. The task is episodic with a time limit of 200 steps. The agents for manipulation are the disembodied manipulator and the default single-arm robot, consisting of a movable base, a Franka arm, and a Panda gripper [8]. The initial pose of agents and the selection of cabinets are randomized by seeds at the start of every episode.

Based on the public-available 25 cabinets that contain drawers in ManiSkill, we randomly split them into a training set of 15 cabinets and a test set of 10 cabinets. In the training process, the agent will not interact or see the test cabinets. We also provide an ablation study of random splits.

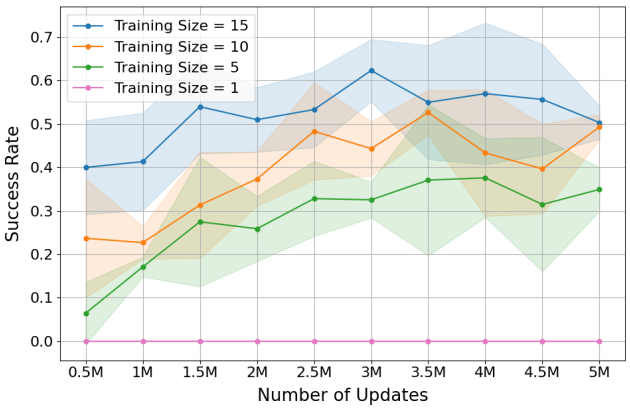
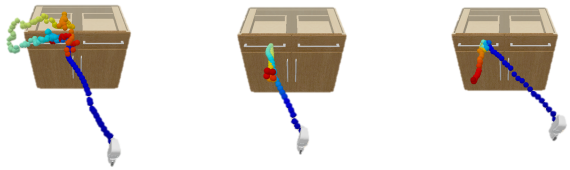


Fig. 5. **Average success rates on unseen cabinets using RL models with varying training set size.** We shows a 95% confidence interval of the mean performance on test cabinets across three random seeds, where we run 100 episodes for each seed. The training set size is the number of cabinets for training.



(a) Failure, step = 200, training size = 5. (b) Success, step = 164, training size = 10. (c) Success, step = 50, training size = 15.

Fig. 6. **Visualization of trajectories on a unseen cabinet using RL models with varying training set size.**

A. Generalizability Evaluation with Varying Training Set

We demonstrate our RL model improving during interaction with more cabinets in the environments (as shown in Fig. 5) by varying different training sizes (i.e., the number of different cabinets for training). We notice that RL converges faster and reaches a higher success rate in the test cabinets when the training size is small. However, these models lack generalizability to novel cabinets that the worse case can be totally unsuccessful. The quantitative results of the training size of 1, 5, 10, 15 in Fig. 5 indicate the continuously evolving process of RL model learning skill dynamics.

Fig. 6 shows the qualitative results by visualizing the disembodied hand trajectories on three test cabinets. The policy models for generating these trajectories are trained with different numbers of cabinets. The model with fewer training cabinets shows worse skill dynamics. For example, we observe that the 5-cabinet model can move forward to right directions of target links, but it either produces confusing motions near the handle or pulls the drawer back and forth. The other models present more reasonable motions and shorter episode lengths. A common failure is that the manipulator stays still or keeps trembling at the target handle. The reason is that there is a local minimum in the reward

function and we believe it can be improved by modifying the reward function or adding auxiliary losses.

TABLE I

TASK SUCCESS RATES WITH DIFFERENT PERCEPTION MODULES

Algorithm	Average Success Rate		Episode Average Length	
Split	Train	Test	Train	Test
BC-robot	0.38	0.13	137	178
BCQ-robot	0.25	0.11	166	184
SAC-robot	0.08	0.03	189	195
Ours-ee-s	0.86	0.63	77	101
Ours-robot-s	0.83	0.61	90	114
Ours-ee-v	0.78	0.55	93	119
Ours-robot-v	0.74	0.51	95	129

B. Benchmark Comparison

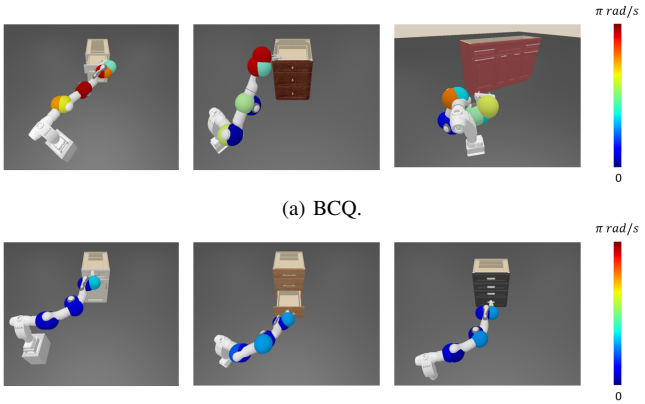
We compare our approach with RL and IL baselines in ManiSkill via two metrics: Average Success Rate and Episode Average Length. The average success rate is calculated by 3 runnings on the test set, where each running uses a different random seed to initialize 100 episodes. In our comparison, all the models are trained to 3M steps.

Our baselines include Behavior Cloning (BC), Batch-Constrained Q-Learning (BCQ), and SAC for whole robot control (SAC-robot). These methods directly predict the joint-space actions to control robots, where BC and BCQ utilize a PointNet+Transformer architecture [9] and SAC uses MLPs. We compare different combinations of modules of our method, where 'ee' and 'robot' represent the disembodied manipulator and the full robot respectively, 's' and 'v' represent states and point clouds as the input respectively.

Table I shows that our method significantly outperforms baselines, indicating better generalizability over novel objects. Our state-based disembodied manipulator shows the best performance on training and test cabinets, which indicates an upper bound of the proposed framework. As for our vision-based agent, it shows a slightly lower performance by adding the whole-body controller and vision module. However, the overall results still surpass baselines. This shows that the decoupling approach is more suitable for generalizable skill learning than baselines.

C. Singularity Estimation of Robot Motion

In addition to the object generalizability, our approach naturally produces compliant and controllable robot actions. The motion compliance, interpretability, and safety are essential for high-DoF and collaborative robots. Fig 7a shows exemplar traces of robots manipulating objects in a singular status. Due to the decreasing rank of the EE's Jacobian matrix, the robot joints are required to accelerate to an extremely high velocity to perform the workspace behaviors. The red balls show the correlated joint velocities hit the limits (π rad/s in our case). A reason for this problem is that high-dimensional actions are challenging to learn by RL (e.g., robots in Fig 7a are controlled by BCQ policy). Therefore, the policy tends to reduce the whole-body DoF causing 'stretching' or 'curling' motions.



(a) BCQ.

(b) Ours.

Fig. 7. **Typical robot motions generated by BCQ (top) and our method (bottom), coloured by joint velocity. Note how BCQ exhibits far higher joint velocities.**

Our approach takes the advantage of robotic physical models and optimization solvers. The whole-body actions well controlled whilst the EE follows the learned policy of the disembodied manipulator. In Fig 7b, our robots apply much lower joint velocities than BCQ-controlled robots but can complete the task with higher success rates.

D. Ablation Study

We perform an ablation study of the random splits of training and test set. Table II shows the average success rate and episode average length of 3 different splits. The overall performances are similar, which demonstrates that our approach is robust to the randomly splitting manner.

TABLE II

TASK SUCCESS RATES WITH RANDOM TRAINING AND TEST SPLITS

Input	Average Success Rate		Episode Average Length	
Split	Train	Test	Train	Test
Set 1-s	0.93	0.59	66	124
Set 2-s	0.86	0.63	77	101
Set 3-s	0.87	0.60	74	115
Set 1-v	0.82	0.54	85	118
Set 2-v	0.78	0.55	93	119
Set 3-v	0.79	0.52	88	125

VI. CONCLUSION

The paper proposed a manipulation learning framework for high-DOF robots where the skill dynamics are decoupled from robot-specific kinematic control policies. In the first stage, a model-free RL agent of the disembodied manipulator is learned from interacting with articulated objects. The QP-based controller then optimizes the high-dimensional joint space actions to approximate the learned skill dynamics by synchronizing the end-effector (EE)'s and disembodied manipulator's trajectories. We evaluate the skill generalizability over objects and achieve an average success rate of 74% in the training set and a 51% in the test set. For future work, we would like to develop the robot generalizability (i.e., generalizing to similar-structured robots) of the learned skill dynamics.

REFERENCES

[1] Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. ManiSkill: Generalizable Manipulation Skill Benchmark with Large-Scale Demonstrations - Sep. *arXiv:2107.14483 [cs]*, November 2021. arXiv: 2107.14483.

[2] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training Home Assistants to Rearrange their Habitat. *arXiv:2106.14405 [cs]*, June 2021. arXiv: 2106.14405.

[3] Kiana Ehsani, Winson Han, Alvaro Herrasti, Eli VanderBilt, Luca Weihs, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. ManipulaTHOR: A Framework for Visual Object Manipulation. *arXiv:2104.11213 [cs]*, April 2021. arXiv: 2104.11213.

[4] Chuang Gan, Jeremy Schwartz, Seth Alter, Damian Mrowca, Martin Schrimpf, James Traer, Julian De Freitas, Jonas Kubilius, Abhishek Bhandwaldar, Nick Haber, Megumi Sano, Kuno Kim, Elias Wang, Michael Lingelbach, Aidan Curtis, Kevin Feigelis, Daniel M. Bear, Dan Gutfreund, David Cox, Antonio Torralba, James J. DiCarlo, Joshua B. Tenenbaum, Josh H. McDermott, and Daniel L. K. Yamins. ThreeDWorld: A Platform for Interactive Multi-Modal Physical Simulation. *arXiv:2007.04954 [cs]*, December 2021. arXiv: 2007.04954.

[5] Bin Fang, Shidong Jia, Di Guo, Muhua Xu, Shuhuan Wen, and Fuchun Sun. Survey of imitation learning for robotic manipulation. *International Journal of Intelligent Robotics and Applications*, 3(4):362–369, December 2019.

[6] Murtaza Dalal, Deepak Pathak, and Ruslan Salakhutdinov. Accelerating Robotic Reinforcement Learning via Parameterized Action Primitives. *arXiv:2110.15360 [cs]*, October 2021. arXiv: 2110.15360.

[7] Karl Pertsch, Youngwoon Lee, and Joseph J. Lim. Accelerating Reinforcement Learning with Learned Skill Priors. *arXiv:2010.11944 [cs]*, October 2020. arXiv: 2010.11944.

[8] Franka Emika. Franka Emika Robot, 2022.

[9] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *arXiv:1612.00593 [cs]*, April 2017. arXiv: 1612.00593.

[10] Oliver Kroemer, Scott Niekum, and George Konidaris. A Review of Robot Learning for Manipulation: Challenges, Representations, and Algorithms. page 82.

[11] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3389–3396, Singapore, Singapore, May 2017. IEEE.

[12] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation. *arXiv:1806.10293 [cs, stat]*, November 2018. arXiv: 1806.10293.

[13] Jun Yamada, Youngwoon Lee, Gautam Salhotra, Karl Pertsch, Max Pflueger, Gaurav S. Sukhatme, Joseph J. Lim, and Peter Englert. Motion Planner Augmented Reinforcement Learning for Robot Manipulation in Obstructed Environments. *arXiv:2010.11940 [cs]*, October 2020. arXiv: 2010.11940.

[14] I.-Chun Arthur Liu, Shagun Uppal, Gaurav S. Sukhatme, Joseph J. Lim, Peter Englert, and Youngwoon Lee. Distilling Motion Planner Augmented Policies into Visual Control Policies for Robot Manipulation. *arXiv:2111.06383 [cs]*, November 2021. arXiv: 2111.06383.

[15] Fei Xia, Chengshu Li, Roberto Martín-Martín, Or Litany, Alexander Toshev, and Silvio Savarese. ReLMoGen: Leveraging Motion Generation in Reinforcement Learning for Mobile Manipulation, 2021.

[16] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987.

[17] Jan Peters and Stefan Schaal. Reinforcement Learning for Operational Space Control. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 2111–2116, 2007.

[18] Josiah Wong, Viktor Makoviychuk, Anima Anandkumar, and Yuke Zhu. OSCAR: Data-Driven Operational Space Control for Adaptive and Robust Robot Manipulation, 2021.

[19] Roberto Martín-Martín, Michelle A. Lee, Rachel Gardner, Silvio Savarese, Jeannette Bohg, and Animesh Garg. Variable Impedance Control in End-Effector Space: An Action Space for Reinforcement Learning in Contact-Rich Tasks, 2019.

[20] Yuke Zhu, Josiah Wong, Ajay Mandlekar, and Roberto Martın-Martın. Robosuite: A Modular Simulation Framework and Benchmark for Robot Learning. page 17.

[21] Jun Nakanishi, Rick Cory, Michael Mistry, Jan Peters, and Stefan Schaal. Operational space control: A theoretical and empirical comparison. *The International Journal of Robotics Research*, 27(6):737–757, 2008.

[22] Hao Shen, Weikang Wan, and He Wang. Learning category-level generalizable object manipulation policy via generative adversarial self-imitation learning from demonstrations. *IEEE Robotics and Automation Letters*, 7(4):11166–11173, 2022.

[23] Kaichun Mo, Leonidas Guibas, Mustafa Mukadam, Abhinav Gupta, and Shubham Tulsiani. Where2Act: From Pixels to Actions for Articulated 3D Objects. *arXiv:2101.02692 [cs]*, August 2021. arXiv: 2101.02692.

[24] Ruihai Wu, Yan Zhao, Kaichun Mo, Zizheng Guo, Yian Wang, Tianhao Wu, Qingnan Fan, Xuelin Chen, Leonidas Guibas, and Hao Dong. VAT-Mart: Learning Visual Action Trajectory Proposals for Manipulating 3D ARTiculated Objects. *arXiv:2106.14440 [cs]*, June 2021. arXiv: 2106.14440.

[25] Ben Eisner, Harry Zhang, and David Held. FlowBot3D: Learning 3D Articulation Flow to Manipulate Articulated Objects, 2022.

[26] Zhenjia Xu, Zhanpeng He, and Shuran Song. UMPNet: Universal Manipulation Policy Network for Articulated Objects. *IEEE Robotics and Automation Letters*, 7(2):2447–2454, 2022.

[27] Mayank Mittal, David Hoeller, Farbod Farshidian, Marco Hutter, and Animesh Garg. Articulated Object Interaction in Unknown Scenes with Whole-Body Mobile Manipulation, 2022.

[28] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[29] Fei Xia, William B. Shen, Chengshu Li, Priya Kasimbeg, Micael Tchapmi, Alexander Toshev, Li Fei-Fei, Roberto Martn-Martn, and Silvio Savarese. Interactive Gibson Benchmark (iGibson 0.5): A Benchmark for Interactive Navigation in Cluttered Environments. *IEEE Robotics and Automation Letters*, 5(2):713–720, April 2020. arXiv: 1910.14442.

[30] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Learning Synergies Between Pushing and Grasping with Self-Supervised Deep Reinforcement Learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4238–4245, October 2018. ISSN: 2153-0866.

[31] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3406–3413, 2016.

[32] Kris Hauser. Klamp't (Kris' Locomotion and Manipulation Planning Toolbox), 2022.

[33] H.J. Ferreau, C. Kirches, A. Potschka, H.G. Bock, and M. Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, 2014.

[34] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A SimulAted Part-based Interactive ENvironment. *arXiv:2003.08515 [cs]*, March 2020. arXiv: 2003.08515.