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 highly 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 in the workspace. It does so by solving a quadratic programming (QP) model with kinematic and robotic singularity constrant based on the physical model of the robot. 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 imitation learning (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 (MPC) or 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], [2], [3], [4], where a visual control policy is learned from interactions through reinforcement learning (RL) or through demonstrations as in imitation learning (IL). However, a common concern of RL is the unproductive exploration in the high-dimensional and continuous action space and state space of whole-body robots. IL, on the other hand, suffers from a distribution shift problem due to the lack of direct and high-quality demonstration of high-DoF robots over numerous scenarios. Hence, it remains very challenging for high-DoF robots to learn complex manipulation skills and further generalize the skill 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. 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 through this decoupling, by making the RL skill task simpler.

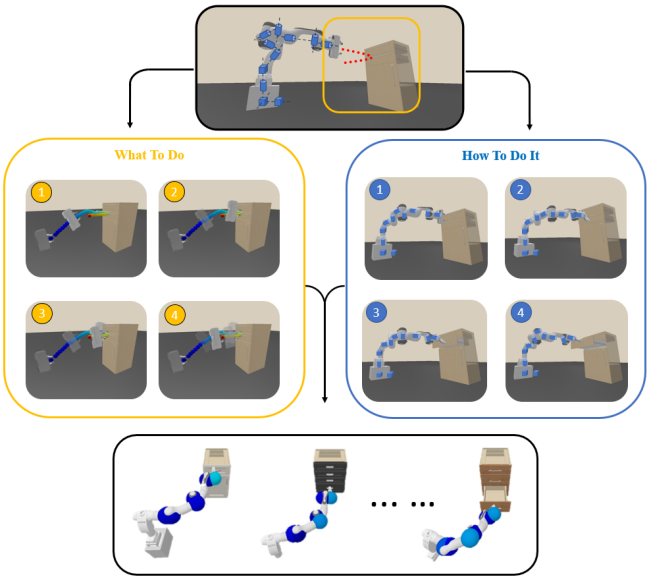


Fig. 1. **Demonstration of decoupling skill dynamics.** The left panel shows how we use reinforcement learning to determine how to control a disembodied or floating end-effector ('what to do'). The right panel shows how we use Quadratic Programming to realize a compliant trajectory and set of joint-torque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

The most similar works to ours are action-primitive methods. Dalal et al.[5] 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. [6] argues that not all actions should be explored with identical probability. They learn priors over skill primitives from offline agent experience. These methods significantly accelerate the RL training process and improve task success rates but not all the manipulation skills can be represented as a series of action primitives. In this

[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

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 process in 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 first calculates the forward kinematics for end-effector (EE) in the workspace based on the robot's full physical model and then formulates a constrained QP with robotic singularity and kinematic constraints. The high-dimensional joint-space actions are produced by: 1) predicting 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 and its physical engine. Our method thus accelerates RL training and convergence, and produces more compliant, smoother and controllable robot motions. In converse, using 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 robot with a Franka arm[7], RGB-D cameras for 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 out-performing 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[8]. However, robotic manipulation of complex objects through control of high-DoF robots remains a challenging problem.

Recent works apply a variety of RL algorithms for robotic manipulation. Gu et al.[9] introduce an off-policy deep Q-Learning network(DQN) for the door opening task. They deploy multiple real robots for parallel training and apply asynchronous policy updating. Similarly, Kalashnikov et al.[10] use DQN for the grasping task. Wu et al. [11] present a policy-gradient SAC method for articulated object manipulation tasks. To improve the pure RL policy, Yamada et al. [12] propose a motion planner augmented RL to avoid collisions in obstructed scenes. In contrast, Liu et al.[13] propose that the motion planner is not robust to state estimation error and they distill the planner into a visual control policy via behavior cloning.

For productive exploration, Dalal et al.[5] manually specify a library of robot action primitives and learn an RL policy to parameterize these primitives. Pertsch et al. [6] learn priors over skill primitives from offline agent experiences. These methods can also be viewed as a fixed-option hierarchical RL [14] where the options are their low-level control policies such as action primitives.

Learning from demonstration (IL) is usually considered as a more sampling-efficient approach[15], [8], [1], but RL and IL are not strictly exclusive. Wang et al. [16] propose a goal-auxiliary policy-gradient RL algorithm using behavior cloning (BC) to initialize the policy model. Kulak et al. [17] jointly learn from the demonstration trajectories and the online experiences of the robot by actively choosing between the two learning modalities.

Another highly related topic is robotic vision for manipulation. Mo et al. [18] introduce an encoder-decoder network to extract pixel-level features and affordance maps and decode the features to action proposals. Similarly, Wu et al. [11] learn the trajectory proposals from point clouds supervised by RL-generated data. These methods show the importance of object representation learning for robotic manipulation and indicate mutual enhancement between RL and vision modules.

III. REINFORCEMENT LEARNING FOR SKILL DYNAMICS

Fig. 2 shows the overall framework of our approach. We first learn the skill dynamics as a disembodied manipulator control policy and then calculate the whole robot's high-dimensional joint actions by QP optimization. We shall discuss the RL and vision module learning in the current section, leaving the discussion of the whole-body controller in section IV. Recent advancements in embodied AI research have greatly improved robotic simulation and provided convenient RL environments for manipulation[2], [4], [19], [3], [20]. In this paper, we use ManiSkill[1] built on SAPIEN[21] as our simulator whose physical engine is NVIDIA PhysX[22]. The joints are controlled by positions or velocities in ManiSkill and then the drives of joints are controlled by torques calculated by inner PIDs of PhysX.

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

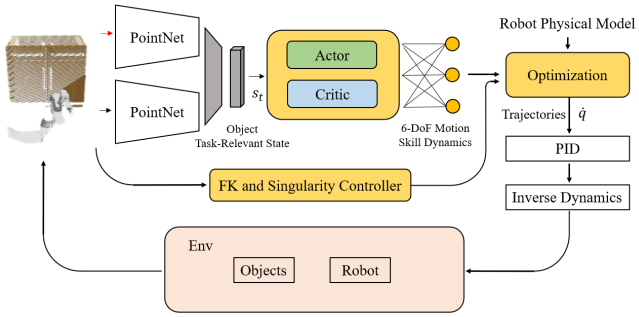


Fig. 2. **Architecture of the proposed system.** From left-to-right: We use an assembly of PointNet[23] modules for perception of static (e.g. size of object) and dynamic (e.g. current position of handle) states. These are input to a SAC RL framework to learn how to control the disembodied end-effector, 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.

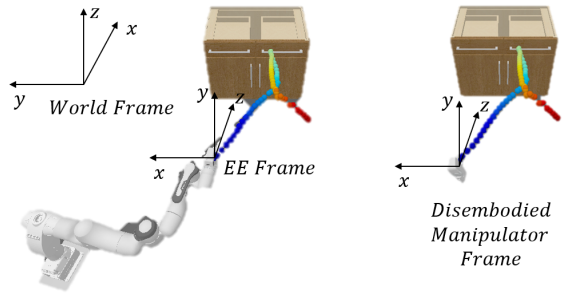


Fig. 3. **Coordinate system of our method.** During the manipulation process, we optimize the robot joint actions to approximate or synchronize EE to the disembodied manipulator's trajectories.

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 $\pi(a|f_s(o))$ where $f_s(o)$ is the states as extracted visual representations and $f(\cdot)$ can be neural networks or hand-crafted operators.

B. Disembodied Manipulator Environment

The agent for learning skill dynamics is a 6-DoF disembodied (or floating/flying) manipulator with 3 virtual prismatic joints and 3 virtual revolute joints. The action space is defined as the 6 velocities of the virtual joints and 2 positions of the finger joints (following the finger control in ManiSkill-Learn[1]). The control signals of the action space are normalized to $(-1, 1)$ for the RL policy. The action in timestep t can be represented as desired values:

$$a_t = [\dot{q}_{0,d}, \dot{q}_{1,d}, \dot{q}_{2,d}, \dot{q}_{3,d}, \dot{q}_{4,d}, \dot{q}_{5,d}, q_{6,d}, q_{7,d}] \quad (1)$$

The state space of a manipulation task that is an interaction-rich process should at least contain robot states, task-relevant object states, and environment states. In our setting, the state space is represented as:

$$s_t = [s_E, s_o, s_{og}, s_r] \quad (2)$$

where the s_E are the environment states, s_o are the general object states, s_{og} are the states of the goal link and its relevant joints and actionable parts, s_r are the robot states. For the cabinet environment of ManiSkill, we design such that $s_o \in \mathbb{R}^{m_1}$ is the cabinet's pose represented by world frame coordinates (x, y, z) and the quaternion for rotation, $s_{og} \in \mathbb{R}^{m_2}$ is the pose of the goal link, joint, and handle, $s_r \in \mathbb{R}^{m_3}$ is the disembodied manipulator's or the whole robot's joint positions, joint velocities and fingers' positions and velocities.

We follow the original design of the dense reward function in the ManiSkill environment which is a continuous and MPC-verified reward formulation. Due to the episodic nature of robotic manipulation tasks, sparse rewards tend to be difficult to learn from[8], [15]. Therefore, the dense reward that is a 3-stage function measures the distance between the manipulator and the target link (e.g., drawer or door), encourages the correct motion of the target link, and rewards the scene for being static after finishing the task [1]. The episode will be terminated after 200 timesteps if the manipulator or the robot fails to complete the manipulation tasks, which is the same time limit setting as ManiSkill.

C. Soft Actor-Critic Algorithm for Manipulation

To learn the skill dynamics, we implement the model-free soft actor-critic (SAC)[14] 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 MLP 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 curve of our RL agent of the disembodied manipulator is shown in Fig. 4, compared with the SAC agent of the 13-DoF whole robot in the same cabinet drawer opening environment. For an identical cabinet, both the robot and the manipulator alone successfully learn the manipulation 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 of training over multiple different cabinets (Fig.5) show that the whole robot configuration almost fails to learn the manipulation skill due to its high-dimensional action space, whilst our manipulator is still capable of learning the skill dynamics over a variety of objects with sufficient task success rates.

D. 3D Visual Perception

The vision module of our work is regarded as a state estimator mapping 3D observations to states s. We utilize an assembly of PointNets[23] as our backbone network that takes masked point clouds as input similar to the benchmark baseline[1]. To achieve a more stable state estimation, we use two PointNet assemblies for the static states (e.g., the

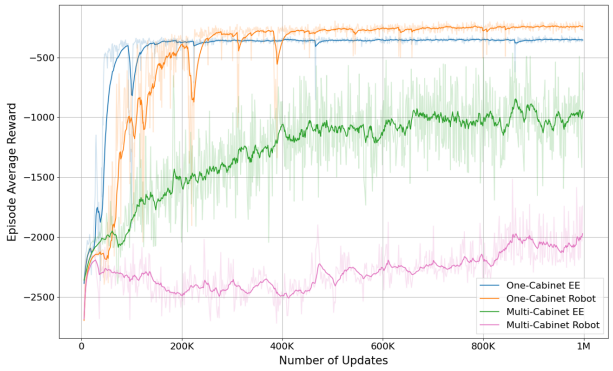


Fig. 4. **Learning curves of RL models.** We compare the learning process of state-based RL agents under two conditions: training with one cabinet or multiple (e.g., 15) different cabinets. For multiple cabinets (i.e., the object generalizability of the training set), the 6-DoF disembodied manipulator converges after around 600K steps while the 13-DoF full robot fails to converge.

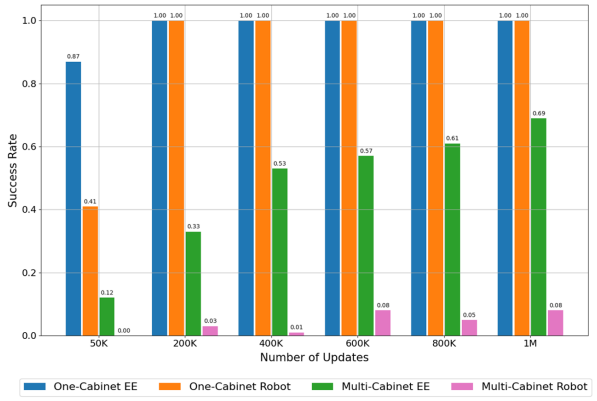


Fig. 5. **Evaluation on training set of SAC models.** We compare the average success rates of state-based RL agents in the training cabinets. The results match the learning curves in Fig. 4

length of drawers) and the 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 implementation such as learning visual representation from interaction (LfI)[24], [25], [26], in future work.

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

$$\mathscr{L}_{vision} = ||s_t - s_t^*||_2 \tag{3}$$

Table.I in sectionV comparing the performances of RL models with ground truth states and vision module 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 the manipulation can be improved by utilizing more accurate 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 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[27] for robotic kinematics calculations and qpOASES[28] 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,

$$x = p(q), q \in \mathbb{R}^n \tag{4}$$

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

$$J(q) = \begin{bmatrix} \partial p_0/\partial q_0 & \partial p_0/\partial q_1 & \dots & \partial p_0/\partial q_{n-1} \\ \partial p_1/\partial q_0 & \partial p_1/\partial q_1 & \dots & \partial p_1/\partial q_n \\ \vdots & \vdots & \ddots & \vdots \\ \partial p_m/\partial q_0 & \partial p_m/\partial q_1 & \dots & \partial p_m/\partial q_n \end{bmatrix} \tag{5}$$

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

$$\dot{x}_{ee} = \begin{bmatrix} \omega \\ v \end{bmatrix} = J(q)\dot{q} \tag{6}$$

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

$$T_{ee} = J(q)\dot{q} - \dot{x}_{ee}^* \tag{7}$$

where $\dot{x}_{ee}^*$ is the desired pose change of robot EE. At the environment timestep 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 the 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.5 $m = 6$ 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,

$$C_{sg} = [c_{base}I_{base}, c_{arm}I_{arm}, c_{finger}I_{finger}] \in \mathbb{R}^{n\times n} \tag{8}$$

where c is the weight parameter and I is 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,

$$z_{res}, q_{ik} = f_{ik}(q_t, x^*_{ee,t+T_{sg}}) \tag{9}$$

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

$$\begin{aligned} c_{base} &= \omega_1 z_{res} + \omega_2(1 - z_{res}) \\ c_{arm} &= \omega_2 z_{res} + \omega_1(1 - z_{res}) \end{aligned} \tag{10}$$

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

$$\begin{aligned} &\underset{\dot{q}_d}{\text{minimize}} \quad \dot{q}_d^T C_{sg} \dot{q}_d \\ &\text{subject to} \\ &\quad J(q)\dot{q}_d = \dot{x}^*_{ee} \\ &\quad \dot{q}_{\min} \le \dot{q}_d \le \dot{q}_{\max} \end{aligned} \tag{11}$$

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 the QP controller.

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 compliancy of our robot motions.

The experiments are performed in the ManiSkill[1] simulated environment built on SAPIEN[21]. In particular, we apply our approach in 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 Franka arm and a Panda gripper[7]. The initial pose of agents and the selection of cabinets are randomized by seeds at the start of every episode.

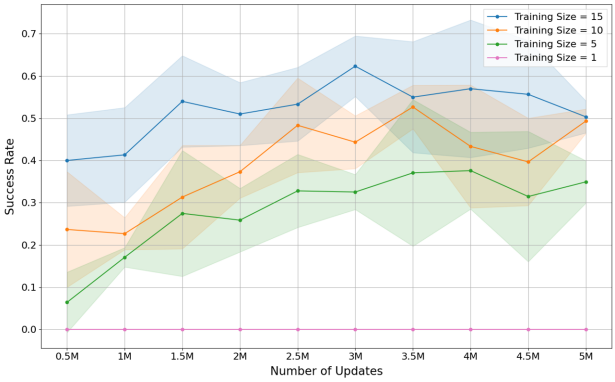


Fig. 6. **Average success rates of RL models with varying training set size.** We shows a 95% confidence interval of the mean performance across three random seeds, where we run 100 episodes for each seed. The training set size is the number of cabinets for training.

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.

The model architecture of our approach is as described in sec.x and the implementation for model training are based on PyTorch. We run all experiments on a desktop PC with NVIDIA RTX 2080Ti.

A. Generalizability Evaluation with Varying Training Set Sizes

We demonstrate our RL model improving during interaction with more cabinets in the environments (as shown in Fig.6) by varying different training sizes (i.e., the number of different cabinets for training). We notice that RL converges faster and reach a higher success rate in the training 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.6 indicate the continuously evolving process of RL model learning skill dynamics.

Fig 7 shows the qualitative results by visualizing the trajectory waypoints of 3 test cabinets RL models with different training sizes. For a fair comparison, the models are trained to the same steps (3M). The model with fewer training cabinets shows worse skill dynamics. For example, we observe that the 5-cabinet model can forward to the right directions of the target link, 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 case 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 fine-tuning the reward function or adding auxiliary losses.

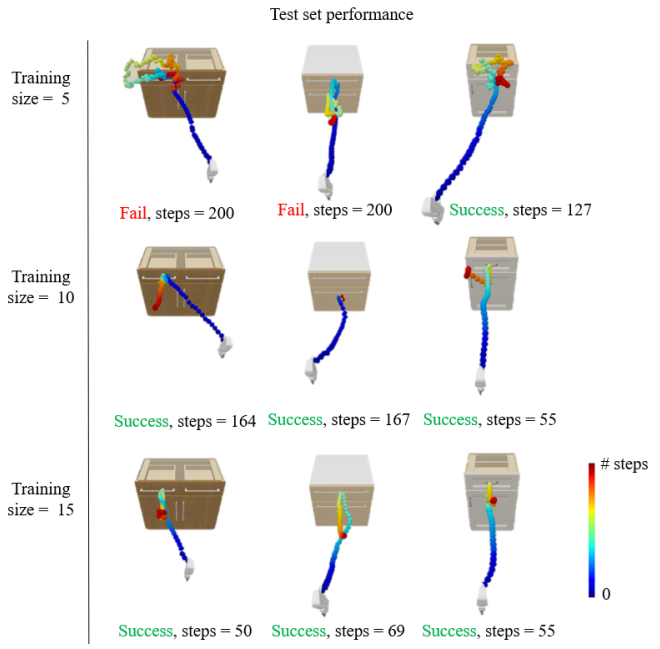


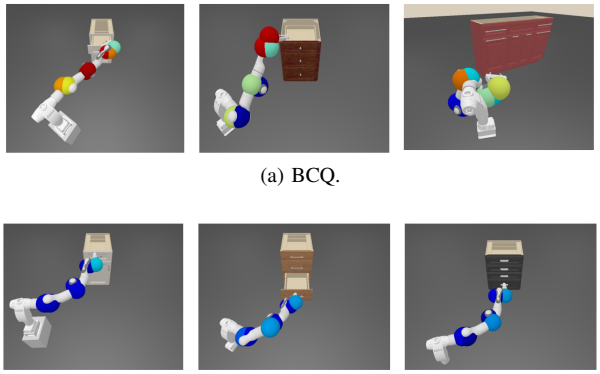
Fig. 7. **Visualization of trajectories predicted by RL models.** We visualize the trajectories for the viewing the improvements of models during training with more cabinets.

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.

The baseline methods contain 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[23] and SAC using MLPs. We compare the 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 in the two metrics, indicating better object generalizability of learned skills. Our agent of the state-based disembodied manipulator shows the best performance on training and test cabinets, which indicates an upper bound of the proposed framework. Further improvement can be using a more informative state representation or a better NN structure for the RL agent. As for our agent of the vision-based robot, it shows a slightly lowering performance by adding the whole-body controller and vision module, but the overall results are still much better than baselines. It demonstrates that the decoupling manner is reasonable and even more suitable for generalizable skill learning than a black-box learning method. There is a lot of room for improvements in vision and control modules, including joint-level torque control by inverse dynamics and more sophisticated networks for object representation learning.



(a) BCQ.

(b) Ours.

Fig. 8. **Typical scenes of BCQ-controlled robots and our method controlled robots**

TABLE I

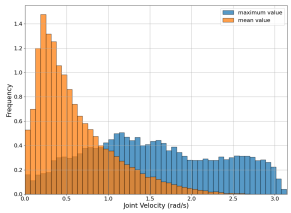
TASK SUCCESS RATES WITH DIFFERENT PERCEPTION MODULES

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

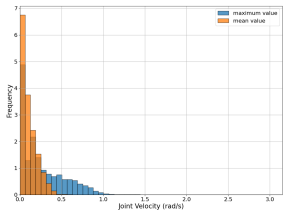
C. Singularity Estimation of Robot Motion

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 8a shows the typical scenes of robots manipulating objects in a singular status. Due to the rank decreasing of 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 reach the limits which is π rad/s in our cases. A reason for this problem is that high-dimensional actions are challenging to learn by RL (e.g., robots in Fig 8a are controlled by BCQ policy). Therefore, the policy tends to reduce the whole-body DoF causing 'stretching' or 'curling' motions.

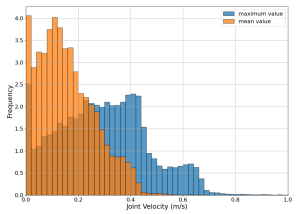
Our approach takes the advantage of robot physical models and optimization solvers. The whole-body actions are under control while the EE follows the learned policy of the disembodied manipulator. In Fig 8b, our robots apply much lower joint velocities than BCQ-controlled robots but can complete the task with higher success rates. A quantitative analysis is shown in Fig. 9. Results present that BCQ required extreme arm and base joint velocities more frequently. In contrast, our approach applies lower arm joint velocities while does not require a fast-moving of the base.



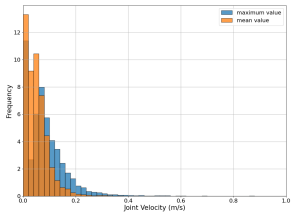
(a) Joints velocities of the robot arm, with BCQ for full robot control.



(b) Joints velocities of the robot arm, with Our method for full robot control.



(c) Joints velocities of the robot base, with BCQ for full robot control.



(d) Joints velocities of the robot base, with Our method for full robot control.

Fig. 9. **Histogram of the joint velocities distribution.**

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. And we shall extend the vision module to an unsupervised learning-from-interaction manner.

VII. ACKNOWLEDGEMENT

We thank Gao Tang for his insightful discussions during the whole research process.

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

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

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

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.

[16] Lirui Wang, Yu Xiang, Wei Yang, Arsalan Mousavian, and Dieter Fox. Goal-Auxiliary Actor-Critic for 6D Robotic Grasping with Point Clouds. page 11.

[17] Thibaut Kulak and Sylvain Calinon. Combining Social and Intrinsically-Motivated Learning for Multi-Task Robot Skill Acquisition. *IEEE Transactions on Cognitive and Developmental Systems*, pages 1–1, 2021.

[18] 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.

[19] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv:1712.05474 [cs]*, March 2019. arXiv: 1712.05474.

[20] Dhruv Batra, Angel X. Chang, Sonia Chernova, Andrew J. Davison, Jia Deng, Vladlen Koltun, Sergey Levine, Jitendra Malik, Igor Mordatch, Roozbeh Mottaghi, Manolis Savva, and Hao Su. Rearrangement: A Challenge for Embodied AI. *arXiv:2011.01975 [cs]*, 2020.

[21] 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.

[22] NVIDIA. NVIDIA PhysX. 2022.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

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

[28] 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.