# Sim-to-Real Reinforcement Learning for Vision-Based Dexterous Manipulation on Humanoids

Toru Lin[1,2], Kartik Sachdev[2], Linxi “Jim” Fan[2], Jitendra Malik[1], Yuke Zhu[2,3]
UC Berkeley[1] NVIDIA[2] UT Austin[3]
https://toruowo.github.io/recipe

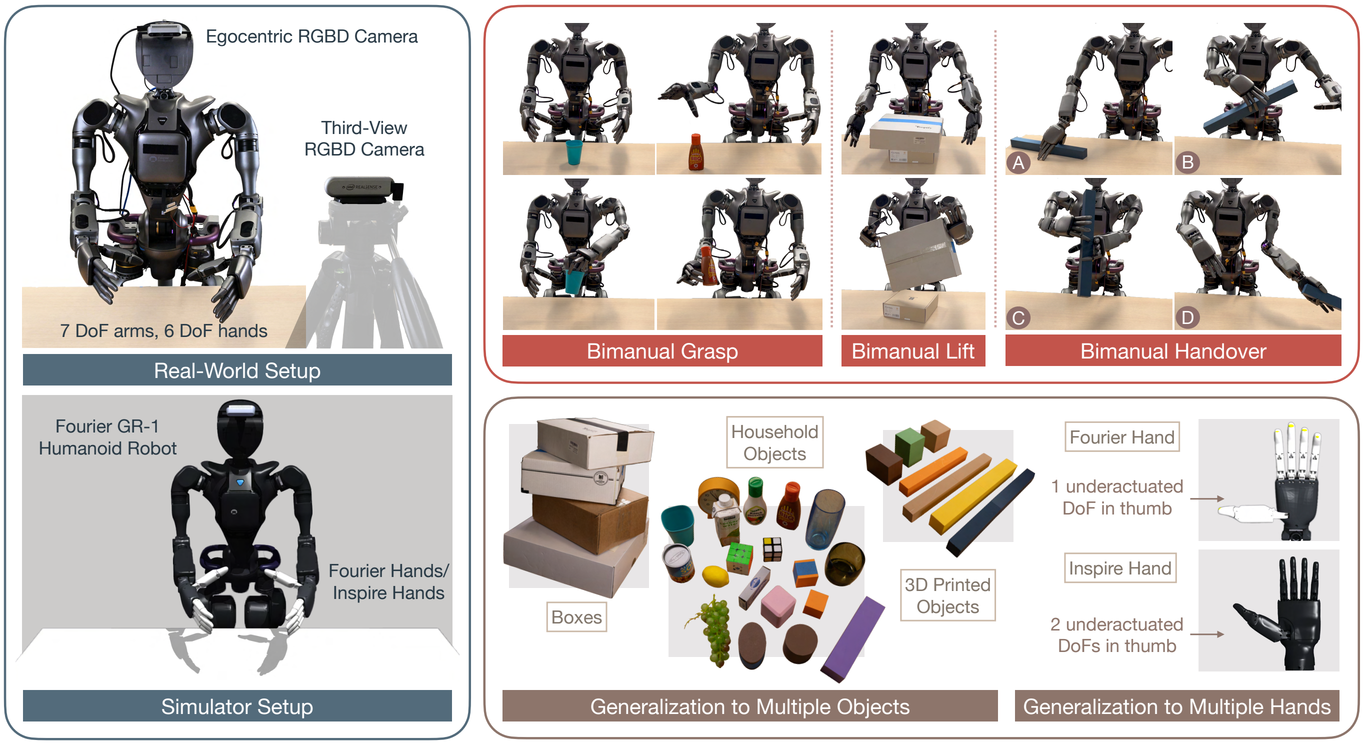

Fig. 1: **Overview.** We train a humanoid robot with two multi-fingered hands to perform a range of contact-rich dexterous manipulation tasks on various objects. Observations are obtained from a third-view camera, an egocentric camera, and robot proprioception. The deployed reinforcement learning policies can adapt to a variety of unseen real-world objects that have varying physical properties (e.g., shape, size, color, material, mass) and remain robust against force disturbances.

*Abstract*—**Reinforcement learning has delivered promising results in achieving human- or even superhuman-level capabilities across diverse problem domains, but success in dexterous robot manipulation remains limited. This work investigates the key challenges in applying reinforcement learning to solve a collection of contact-rich manipulation tasks on a humanoid embodiment. We introduce novel techniques to overcome the identified challenges with empirical validation. Our main contributions include an automated real-to-sim tuning module that brings the simulated environment closer to the real world, a generalized reward design scheme that simplifies reward engineering for long-horizon contact-rich manipulation tasks, a divide-and-conquer distillation process that improves the sample efficiency of hard-exploration problems while maintaining sim-to-real performance, and a mixture of sparse and dense object representations to bridge the sim-to-real perception gap. We show promising results on three humanoid dexterous manipulation tasks, with ablation studies on each technique. Our work presents a successful approach to learning humanoid dexterous manipulation using sim-to-real reinforcement learning, achieving robust generalization and high performance without the need for human demonstration.**

## I. Introduction

Deep reinforcement learning (RL) has delivered a number of impressive results during recent years, covering a diverse range of application domains: classical board games [48], competitive multiplayer online games [6, 56], large language models [1, 14], real-world robotic locomotion [21, 26], autonomous drone racing [24] — to name a few. These accomplishments have not only showcased RL’s potential to achieve or surpass human-level performance across various tasks but also highlighted its distinctive ability to scale and generalize via autonomous exploration. Such inherent characteristics position RL as a performant and long-term approach to tackling tasks that are difficult to solve with human priors or demonstrations.

Motivated by its potential, we explore RL to address challenging dexterous manipulation tasks from vision. The successes that deep RL has produced in this problem domain remain limited so far. Previous works have demonstrated highly dexterous manipulation capabilities that could not be simply programmed or teleoperated by humans [2, 17, 31]. However, these approaches are often tailored to a single manipulation skill, limiting their broad applicability.

What prevents RL from being more generally applicable to vision-based dexterous manipulation? We first investigate this by identifying the inherent properties of dexterous manipulation that differentiate this application domain from others. Then, we examine how these properties contribute to challenges in applying RL algorithms and develop a collection of novel techniques to address the challenges. Putting together our experiences and techniques, we outline a recipe for applying sim-to-real RL to vision-based humanoid manipulation tasks and show promising results. Below, we articulate the key challenges and our strategies to tackle them.

**Challenge in environment modeling.** The first challenge in applying RL to dexterous manipulation lies in the difficulty (or impossibility) of matching a simulated environment with the real environment. While real-world RL circumvents this problem, training with physical hardware is highly demanding regarding hardware quality, maintenance support, controller robustness, and safety. With a system as high-dimensional as a humanoid with multi-fingered hands, real-world exploration becomes even less tractable. In contrast, simulations offer unlimited chances of trial and error in a virtual sandbox, motivating the development of sim-to-real RL approaches. While previous successes in RL-based locomotion [16, 21, 26, 43] are encouraging, we observe that previous successes in dexterous manipulation involve much more laborious real-to-sim engineering efforts that are task-specific or hardware-specific [2, 17, 31]. To better align simulation with the real world, we propose an automated real-to-sim tuning module that substantially reduces the engineering effort required for the environment modeling gap.

**Challenge in reward design.** While the reward function is commonly used as a general interface for specifying a task to train RL policies, it is notoriously hard to design generalizable rewards for manipulation tasks, especially for those that are contact-rich or long-horizon. Prior work often resorts to hand engineering based on the knowledge of human experts [40, 61], which has limited scalability in the long run. This challenge differentiates manipulation from locomotion, where many tasks of interest can be specified with variations of the reward for a single "walking" task. We propose a general principle to design rewards for dexterous manipulation tasks: disentangle a full task into intermediate "contact goals" and "object goals". We use a novel keypoint-based state representation to specify contact goals. Following our reward design techniques, a task as long-horizon and contact-rich as bimanual handover can be learned with RL *tabula rasa*.

**Challenge in policy learning.** A well-defined reward function does not guarantee the successful learning of RL policies due to the sample complexity and reward sparsity of exploring in a high-dimensional space. The variety and complexity of contact patterns in dexterous manipulation with multi-fingered hands further exacerbate the problem. Although unsupervised methods [7, 30, 39] have been proposed to encourage exploration by favoring novel state visitations, they do not fundamentally reduce the difficulty of hard-exploration problems. We tackle this challenge by introducing two practical techniques: (1) initializing tasks with task-aware hand poses; (2) breaking down hard exploration problems into sub-tasks with much-reduced dimensionality, training expert policies on the sub-tasks, then distilling them into a generalist policy for the full task. We experimentally verify that these techniques improve sample efficiency of learning and study how different divide-and-conquer schemes vary in effectiveness.

**Challenge in object perception.** Compared to other robotic tasks, object perception is particularly important for manipulation because the task is inevitably coupled with interaction with objects. Object perception is a long-standing challenge because the variety of objects is uncountable in shapes, sizes, masses, colors, textures, and many other properties. Research in applying sim-to-real RL to dexterous manipulation is bottlenecked by this dilemma — while object representations that are more expressive and information-dense can improve dexterity and capability of the learned policy, they also present a larger sim-to-real gap. To overcome this challenge, we propose to use a mixture of low-dimensional and high-dimensional object representations, with modality-specific data augmentation on the high-dimensional features to reduce the sim-to-real perceptual gap. We systematically study how this combination could help achieve a good balance between learning dexterous manipulation policy and reliably transferring the policy onto real robot hardware.

The strategies we outline above form a complete recipe of sim-to-real RL for vision-based dexterous manipulation. We show successful results of learning a collection of three dexterous manipulation tasks on humanoids and conduct systematic ablation studies.

## II. Background

### A. Deep Reinforcement Learning Applications to Robotics

The successes of deep RL across a wide range of applications [1, 6, 14, 21, 24, 26, 48, 56] have sparked lots of excitement in recent years. However, works over the years have identified brittleness with this paradigm, most notably the sensitivity to hyperparameters [19] and questionable reproducibility [23] due to the high variance intrinsic to RL algorithms.

Among the open problems in RL, the most important and long-standing is exploration. In supervised learning, it is often assumed that data is given. In RL, however, agents need to collect their own data and update their policy based on the collected data. The problem of *how* data is collected is known as the exploration problem. Real-world robotics, with high-dimensional observations and dynamics and often sparse rewards, present a particularly challenging set of hard exploration problems for RL. While there have been works that algorithmically scale exploration to high-dimensional inputs by encouraging visitation to novel states [4, 7, 30, 38, 39, 50, 53],

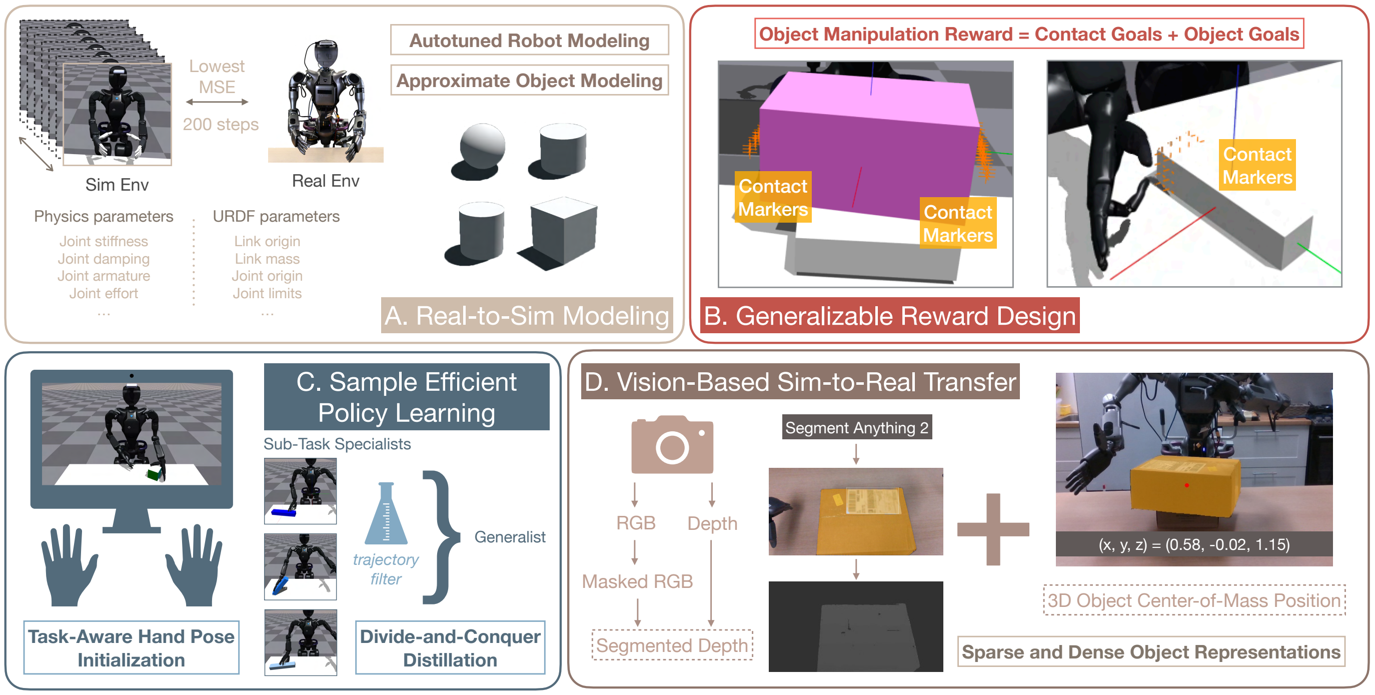

Fig. 2: **A sim-to-real RL recipe for vision-based dexterous manipulation.** We close the environment modeling gap between simulation and the real world through an automated real-to-sim tuning module, design generalizable task rewards by disentangling each manipulation task into contact states and object states, improve sample efficiency of dexterous manipulation policy training by using task-aware hand poses and divide-and-conquer distillation, and transfer vision-based policies to the real world with a mixture of sparse and dense object representations.

they do not fundamentally resolve the exploration challenge. Additionally, applying RL to solve real-world robotics also reveals important challenges that standard benchmarks in RL [5, 54] fail to capture: (1) the lack of fully or accurately modeled environments; (2) the lack of well-defined reward functions for tasks of interest.

Past works in the intersection of robotics and RL have proposed various practical techniques to alleviate these problems, such as learning from human motion data or teleoperated demonstrations [9, 45, 60, 65], real-to-sim techniques to model object and visual environments [2, 16, 17, 31, 55] and more principled ways to design rewards [37, 62]. While some of them overfit to specific tasks and settings, they point to promising directions upon which this work builds.

### B. Vision-Based Dexterous Manipulation on Humanoids

**Imitation learning and classical approaches.** Innovations in teleoperation [10, 32, 58, 63] and learning from demonstrations [11, 28] have brought about many recent advances in vision-based dexterous manipulation [10, 28, 32, 64]. However, in practice, onboarding teleoperators to collect high-quality dexterous manipulation data remains costly, and the performance scaling with data purely collected from real-world teleoperation [27, 29, 64] suggests that the cost to reach human-level performance could be prohibitively large.

**Reinforcement learning approaches.** A number of existing works have successfully applied RL to solve dexterous manipulation problems with multi-fingered hands, but either assume a single-hand setup [2, 8, 17, 35, 42, 49, 57] or do not use pixel inputs as object representation [9, 20, 31]. Moreover, most of the existing works focus on a single manipulation skill, including in-hand reorientation [2, 17, 42, 57], grasping [35, 49], twisting [31], and dynamic handover [20]. The closest to our work is Chen et al. [9], but their method relies on human hand motion capture data to learn a wrist controller rather than learning the full hand-arm joint control from scratch. In addition, existing works often focus on hardware whose models in physics simulation have been more extensively tested. Our work is the first to show successful sim-to-real RL transfer of vision-based dexterous manipulation policies to a novel humanoid hardware with multi-fingered hands.

## III. Challenges and Approaches

In Section I, we identify four areas of challenges in applying sim-to-real RL to dexterous manipulation and briefly describe our strategies to tackle the challenge in each area. Below, we describe our specific approaches in detail. Figure 2 shows an overview of the challenges and approaches.

### A. Real-to-Sim Modeling

Simulators offer unlimited trial-and-error chances to perform the exploration necessary for RL. However, whether policies learned in simulation can be reliably transferred to the real world heavily depends on the faithfulness of modeling — both the robot itself and the environment. When applying sim-to-real RL to solve dexterous manipulation, this real-to-sim modeling problem is further exacerbated by the necessity to model objects, which have great variability and whose full physical properties cannot be easily quantified. Even when one assumes that the ground-truth physical parameters are known, quantitatively matching the simulation with the real world is hard: due to the limitations of physics engines, the same values for physical constants in simulation and the real world do not necessarily correspond to identical kinematic and dynamic relationships.

**Autotuned robot modeling.** While robot manufacturers are often able to provide proprietary model files for their robot hardwares, the models mostly serve a starting reference for robot real-to-sim effort rather than ground truth models that can be used without modifications. Empirical solutions to increase modeling accuracy range from hand-tuning the robot model constants and simulatable physical parameters [2] to reformulating specific kinematic structures (e.g., four-bar linkage) in the simulator of choice [44]. This is a laborious process as there is no "ground truth" pairing between the real world and the simulated world. We propose a practical technique to speed up this real-to-sim modeling process via an "autotune" module. The autotune module enables rapid calibration of simulator parameters to match real robot behavior by automatically searching the parameter space to identify optimal values for both simulator physics and robot model constants in under four minutes (or 2000 simulated steps in 10 Hz). We illustrate the module in Figure 2A and Algorithm 1. The module operates on two parameter types: simulator physics parameters affecting kinematics and dynamics, as well as robot model constants from the URDF file (including link inertia values, joint limits, and joint/link poses). The calibration process begins by initializing multiple simulated environments using randomly sampled parameter combinations from the parameter space, bootstrapped from the manufacturer's robot model file. It then executes $N$ calibration sequences consisting of joint position targets on both the real robot hardware (single run) and all simulated environments in parallel. By comparing the tracking error between each simulated environment and the real robot when following the same joint targets, the module selects the parameter set that minimizes the mean squared error in tracking performance. This approach eliminates iterative manual tuning by requiring only one set of calibration runs on the real robot, automatically optimizing traditionally hard-to-tune URDF parameters, and supporting parallel evaluation of multiple parameter combinations. The method's generality allows it to tune any exposed simulator or robot model parameter that affects kinematic behavior.

**Approximate object modeling.** As demonstrated in previous works [31, 41], modeling objects as primitive shapes like cylinders with randomized parameters is sufficient for sim-to-real transferrable dexterous manipulation policies to be learned. Our recipe adopts this approach and finds it effective.

**Algorithm 1** Real-to-Sim Autotune Module

```
Require:
1:  $E$ : Set of environment parameters to tune
2:  $N$ : Number of calibration action sequences
3:  $R$ : Real robot hardware environment
4:  $M$ : Initial robot model file
5:  procedure AUTOTUNE($E, N, R, M$)
6:      $P \leftarrow$ InitializeParameterSpace($E, M$)    ▷ Initialize from model
7:      $S \leftarrow \{\}$    ▷ Set of simulated environments
8:      for $i \leftarrow 1$ to $K$ do    ▷ $K$ is population size
9:          $p_i \leftarrow$ RandomSample($P$)
10:         $S_i \leftarrow$ CreateSimEnvironment($p_i$)
11:         $S \leftarrow S \cup \{S_i\}$
12:     end for
13:     $J \leftarrow$ GenerateJointTargets($N$)    ▷ Joint target sequences
14:     $R_{track} \leftarrow$ GetTrackingErrors($R, J$)    ▷ Real tracking
15:     $best\_params \leftarrow$ null
16:     $min\_error \leftarrow \infty$
17:     for $S_i \in S$ do
18:         $S_{track} \leftarrow$ GetTrackingErrors($S_i, J$)
19:         $error \leftarrow$ ComputeMSE($S_{track}, R_{track}$)
20:         if $error < min\_error$ then
21:             $min\_error \leftarrow error$
22:             $best\_params \leftarrow$ GetParameters($S_i$)
23:         end if
24:     end for
        return $best\_params$
25: end procedure
```

## B. Generalizable Reward Design

In the standard formulation of RL [51], the reward function is a crucial element within the paradigm because it is solely responsible for defining the agent's behavior. Nevertheless, the mainstream of RL research has been preoccupied with the development and analysis of learning algorithms, treating reward signals as given and not subject to change [13]. As tasks of interest become more general, designing reward mechanisms to elicit desired behaviors becomes more important and more difficult [12] — as is the case of applications to robotics. When it comes to dexterous manipulation with multi-fingered hands, reward design becomes even more difficult due to the variety of contact patterns and object geometries.

**Manipulation as contact and object goals.** From a wide variety of human manipulation activities [15], we observe a general pattern in dexterous manipulation: each motion sequence to execute a task can be defined as a combination of hand-object contact and object states. Building on this intuition, we propose a general reward design scheme for even long-horizon contact-rich manipulation tasks. For each task of interest, we first break it down into an interleaving sequence of contact states and object states. For example, the handover task can be broken down into the following steps: (1) one hand contacting the object; (2) the object being lifted to a position near the other hand; (3) the other hand contacting the object; (4) object being transferred to the final goal position. The reward can then be defined based on solely the "contact goals" and "object goals": each contact goal can be specified by penalizing the distance from fingers to desirable contact points or simply the object's center-of-mass position; each object goal can be specified by penalizing the distance from its current state (e.g., current *xyz* position) to its target state (e.g., target *xyz* position). To reduce the difficulty of specifying contact goals, we propose a novel keypoint-based technique as follows: for each simulated asset, we procedurally generate a set of "contact stickers" attaching to the surface of the object, where each sticker represents a potentially desirable contact point. The contact goal, in terms of reward, can then be specified as

$$r_{\text{contact}} = \sum_i \left[ \frac{1}{1+\alpha d(\mathbf{X}^L, \mathbf{F}_i^L)} + \frac{1}{1+\beta d(\mathbf{X}^R, \mathbf{F}_i^R)} \right], \tag{1}$$

where $\mathbf{X}^L \in \mathbb{R}^{n\times 3}$ and $\mathbf{X}^R \in \mathbb{R}^{m\times 3}$ are the positions of contact markers specified for left and right hands, $\mathbf{F}^L \in \mathbb{R}^{4\times 3}$ and $\mathbf{F}^R \in \mathbb{R}^{4\times 3}$ are the position of left and right fingertips, $\alpha$ and $\beta$ are scaling hyperparameters, and $d$ is a distance function defined as

$$d(\mathbf{A}, \mathbf{x}) = \min_i \|\mathbf{A}_i - \mathbf{x}\|_2. \tag{2}$$

We show a visualization of contact markers in Figure 2B, and experimental results on their effectiveness in Section IV.

## C. Sample Efficient Policy Learning

Due to the sample complexity and reward sparsity in exploring a high-dimensional space — especially on a humanoid embodiment with multi-fingered hands — policy learning can take a prohibitively long time, even with a well-defined reward function. We propose two techniques that more effectively improve the sample efficiency of policy learning: (1) initializing tasks with task-aware hand poses; (2) dividing a challenging task into easier sub-tasks, then distilling the sub-task specialists into a generalist policy.

**Task-aware hand poses for initialization.** We reduce the exploration challenge by collecting task-aware hand pose data from humans. This can be done by connecting any teleoperation system for bimanual multi-fingered hands to the simulator of choice. The collected states, including object poses and robot joint positions, are then randomly sampled as task initialization states in simulation. Distinct from prior works that require full demonstration trajectories [3], we find that teleoperators need not accomplish the task and only need to "play around" with the task goal in mind while environmental states are collected. The approach massively reduces the time needed for teleoperation since human operators do not need to spend time "ramping up" to collect high-quality data. In our experiments, each task requires less than 30 seconds for sufficient amount of hand pose data to be collected.

**Divide-and-conquer distillation.** Previous methods to improve sample efficiency of policy learning mostly focus on exploring the state space more efficiently [7, 30, 39, 52]. However, these methods do not reduce the difficulty of the exploration problem fundamentally: the probability of receiving learning signals from exploring the "right" states remains the same. Following this observation, we reason that an easier way to overcome the exploration problem in sparse reward settings is to break down the explorable state space itself. For example, a multi-object manipulation task can be divided into multiple single-object manipulation tasks. After dividing a complex task into easier sub-tasks, we can train specialized policies for each sub-task and distill them into a generalist policy. Another benefit of this approach is that we can flexibly filter out trajectory data from the sub-task policies based on their optimality and only retain high-quality samples for training. This effectively brings RL closer to learning from demonstrations, where the sub-task policies act as teleoperators for task data collection in the simulation environment, and the generalist policy acts as a centralized model trained from curated data.

## D. Vision-Based Sim-to-Real Transfer

Transferring a policy learned in simulation to the real world is challenging because of the sim-to-real gap. In the case of vision-based dexterous manipulation, the sim-to-real gap stems from both dynamics and visual perception — both are challenging open research problems to solve. We outline two key techniques we employ to reduce the gap.

**Mixing object representations.** Object perception is crucial for dexterous manipulation because the task is inevitably coupled with object interaction. Prior works that show successful sim-to-real transfer of manipulation policies have explored a wide range of object representations, including (in order of increasing dimensionality and complexity) 3D object position [31], 6D object pose [2], depth [35, 42], point cloud [33], and RGB images [17]. There is a delicate trade-off between using these different object representations: while higher-dimensional representations encode richer information about the object, the larger sim-to-real gap in those data modalities makes the learned policy harder to be transferred; on the other hand, it is harder to learn optimal policies with lower-dimensional object representations because of the limited amount of information. We, therefore, propose a combination of both types of object representations to balance the trade-offs: a low-dimensional 3D object position and a high-dimensional depth image. Importantly, the 3D object position is obtained from a third-view camera to ensure the object is also in camera view and its noisy position can be consistently tracked. The depth image complements with information on object geometry. We provide more empirical validation on the effectiveness of this approach in Section IV.

**Domain randomization for dynamics and perception.** We apply a wide range of domain randomizations to ensure reliable sim-to-real transfer. We list the details in Appendix C.

## IV. Experiments

Our proposed approaches form a general recipe that allows for the practical application of RL to solve dexterous manipulation with humanoids. In this section, we show experimental results of task capabilities and ablation studies of each proposed technique. Videos can be found on project website.

### A. Real-World and Simulator Setup

We use a Fourier GR1 humanoid robot with two arms and two multi-fingered hands. Each arm has 7 degrees of freedom (DoF). For most experiments, we use the Fourier hands, each of which has six actuated DoFs and five underactuated DoFs. To show cross-embodiment generalization, we include results on the Inspire hands, each with 6 actuated DoFs and six underactuated DoFs. The hardware has substantially different mass properties, surface frictions, finger and palm morphologies, and thumb actuations. Figure 1 shows a visualization of both robot hands. We use the NVIDIA Isaac Gym simulator [36].

**Perception.** As outlined in Section III-D, we use a combination of dense and sparse object features for policy learning in both simulation and real-world transfer. In the real world, we set up an egocentric-view camera by attaching a RealSense D435 depth camera onto the head of a humanoid robot and a third-view camera by putting another RealSense D435 depth camera on a tripod in front of the robot (illustrated in Figure 1). In simulation, we similarly set up an egocentric-view camera and a third-view camera by calibrating the camera poses against the real camera poses. The *dense* object feature is obtained by directly reading depth observations from the egocentric-view camera. The *sparse* feature is obtained by approximating the object's center-of-mass from the third-view camera, using a similar technique as in Lin et al. [31]. As illustrated in Figure 2, we utilize the Segment Anything Model 2 (SAM2) [46] to generate a segmentation mask for the object of interest at the first frame of each trajectory sequence and leverage the tracking capabilities of SAM2 to track the mask throughout all remaining frames. To approximate the 3D center-of-mass coordinates of the object, we calculate the center position of their masks in the image plane, then obtain noisy depth readings from a depth camera to recover a corresponding 3D position. The perception pipeline runs at 5 Hz to match the neural network policy's control frequency.

### B. Task Definition

**Grasp-and-reach.** In this task, the robot needs to use one hand to grasp a tabletop object, lift it up, and move it to a goal position. At task initialization, a scripted visual module determines which hand is closer to the object and instructs the robot to use that hand. The test objects have varying geometric shapes, masses, volumes, surface frictions, colors, and textures; a visualization of all objects can be found in Figure 1. For each trial, we vary the initial position and orientation of objects, as well as the goal position.

**Box lift.** In this task, the robot needs to lift a box that is too large to be grasped with a single hand. Box colors, sizes, and masses are varied. For each trial, we randomize the boxes' initial position and orientation about the vertical axis.

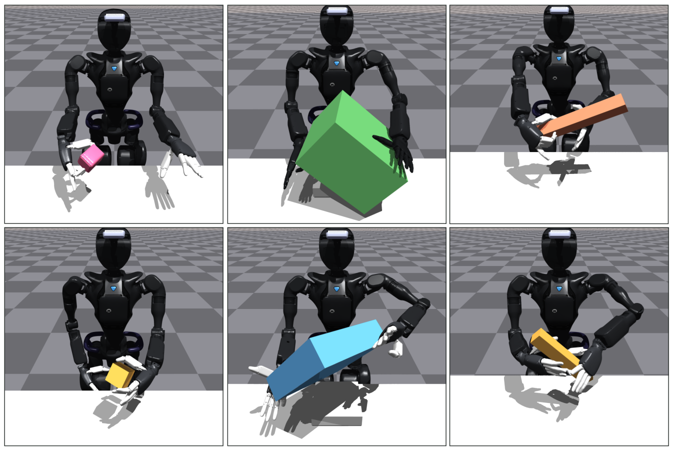
Fig. 3: **Policies learned in simulation.** Left: grasp; middle: box lift; right: bimanual handover (right-to-left, left-to-right).

| Autotune MSE | Lowest | Median | Highest |
|---|---|---|---|
| Grasp Success | 8 / 10 | 3 / 10 | 0 / 10 |
| Reach Success | 7 / 10 | 3 / 10 | 0 / 10 |

TABLE I: **Lower MSE from autotune correlates with higher sim-to-real success rate.** For each set of modeling parameters, we test the sim-to-real transfer performance of 10 policy checkpoints (trained identically except for random seed). We evaluate success rate by stages on the `grasp-and-reach` task, and observe a correlation between lower MSE measured by autotune module and higher sim-to-real transfer success rate.

**Bimanual handover.** In this task, the robot needs to grasp a block object from one side of the table with one hand — that is too far to reach for the other hand — and hand the object over to the other hand. Objects include blocks of varying colors, dimensions, and masses. We vary the initial position and orientation of blocks in each trial.

### C. Evaluation of Real-to-Sim Modeling

**Effectiveness of autotuned robot modeling.** We apply the autotune module outline in Section III-A to find the optimal parameter set for robot modeling. To evaluate its effectiveness, we compare the sim-to-real transfer success rates of three sets of policy checkpoints. All policies are trained with identical task and RL settings, but each set of policies uses robot modeling parameters that achieve different MSE from autotune, ranging from lowest (i.e., smallest real-to-sim gap) to highest (i.e., highest real-to-sim gap). The quantitative results in Table I indicate that autotuned robot modeling improves sim-to-real transfer. Additionally, in our video, we show qualitative results of successful sim-to-real transfer of `grasp-and-reach` policies to the Inspire hands, demonstrating the generalizability of our autotune module.

**Effectiveness of approximate object modeling.** Empiri-

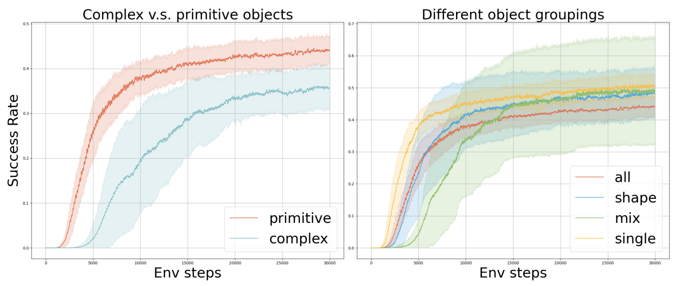

Fig. 4: **Training `grasp-and-reach` policy with different object sets.** Each curve is computed from the statistics of 10 training runs with different random seeds. Left: training with complex objects v.s. simple geometric primitive objects. Right: training with differently grouped geometric objects.

cally, we find that modeling objects as primitive geometric shapes (cylinders, cubes, and spheres) strikes a good balance between training efficiency and sim-to-real transferability. In Figure 4 (left), we compare the training curves of `grasp-and-reach` policies with primitive shapes against those with complex shapes, and the former setting is more sample efficient. More importantly, policies trained with randomized primitive shapes can also generalize to a variety of unseen objects, as shown in our video.

## D. Evaluation of Reward Design

**Task capabilities.** With our proposed reward design principle, a wide range of long-horizon contact-rich tasks can be accomplished with pure reinforcement learning, as shown in Figure 3 and our video. The learned policies exhibit a high degree of dexterity and robustness against random force disturbances.

**Effectiveness of contact-based rewards.** In Figure 5, we visualize different contact behaviors that emerged from different placements or contact markers, using the `box lift` task as an example. We find that contact behaviors align closely with the contact positions specified, showing the effectiveness of using contact markers to specify contact goals.

## E. Evaluation of Policy Learning

**Effectiveness of task-aware hand pose initialization.** In Table II, we compare the percentage of successfully trained policies for each task with and without task-aware hand pose initialization. The empirical results show that having human priors upon initialization can greatly improve the exploration efficiency of hard RL tasks.

**Divide-and-conquer distillation.** We evaluate our divide-and-conquer distillation approach through two ablation studies. First, we study the effect of divide-and-conquer granularity on training efficiency. Specifically, we experiment with dividing a multi-object `grasp-and-reach` task into sub-tasks that handle different numbers of objects. Starting from a total of 10 objects, we experiment with three task designs: (1) training with all objects in one policy (*all*); (2) training with three groups of similarly shaped objects in three policies (*shape*); (3) training with three groups of differently shaped objects

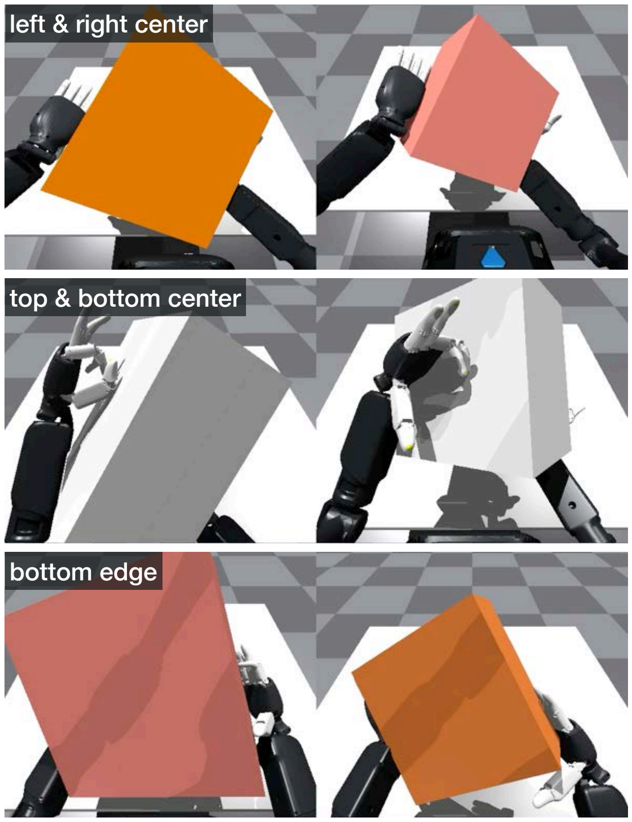

Fig. 5: **Different contact patterns emerge from different placements of contact markers.** Top: contact markers on the left and right side centers; middle: markers on the top and bottom side centers; bottom: markers on the bottom side edges.

in three policies (*mix*); (4) training with ten single-object policies (*single*). As shown in Figure 4, sample efficiency is highest for *single*, followed by *shape*, *all*, and *mix*. There is also a noticeable difference in the average success rates of each task, which could be explained as an indicator of task difficulty. Interestingly, while training with a reduced object set all reaches the same performance, training all objects in one policy shows a consistently lower performance. Second, we study the sim-to-real transfer success rate of each type of policy. Over 30 trials of each policy on an in-distribution object, we find that sim-to-real performance of the `mix` policy is the highest (90.0%), followed by `shape` (63.3%), `single` (40.0%), and `all` (23.3%). Based on the qualitative behavior of policies, we hypothesize that the low success rates of `mix` and `single` policies stem from overfitting to specific geometries, and that of `all` policy correlates with its worse performance during RL training. These results suggest that divide-and-conquer distillation helps achieve a good balance between policy training and sim-to-real transfer performance.

## F. Evaluation of Vision-Based Sim-to-Real Transfer

**Effectiveness of mixing object representations.** We investigate the effect of using different object representations and show the sim-to-real transfer comparisons in Table III. These results suggest combining dense object representation (segmented depth image) and sparse object representation (3D

| % Success | Grasping | Lifting | Handover |
|---|---|---|---|
| with Human Init | 80% | 90% | 30% |
| w/o Human Init | 60% | 90% | 0% |

TABLE II: **Initializing with human data.** Correlation between the percentage of successfully learned task policies and whether human play data is used for initialization. We define *successfully learned* policies as those that achieve over 60% episodic success during evaluation. For each task and each initialization setting, we test with 10 random seeds.

| Task | Grasping | Lifting | HandoverA | HandoverB |
|---|---|---|---|---|
| **Depth + Pos** | | | | |
| Pickup | 10 / 10 | 10 / 10 | 10 / 10 | 10 / 10 |
| Task Success | 10 / 10 | 10 / 10 | 9 / 10 | 5 / 10 |
| **Depth Only** | | | | |
| Pickup | 2 / 10 | 0 / 10 | 0 / 10 | 0 / 10 |
| Task Success | 2 / 10 | 0 / 10 | 0 / 10 | 0 / 10 |

TABLE III: **Comparison of sim-to-real transfer performance between depth-and-position policy and depth-only policy.** We separate the `bimanual handover` task into two columns due to its longer horizon. The pickup success is an intermediate success metric that measures how often the hands successfully pick up the object of interest. We find that combining low-dimensional representation (3D object position) with depth enables easier sim-to-real transfer.

object center-of-mass position) improves sim-to-real transfer. Notably, the gap between the success rate of the depth-and-position policy and that of the depth-only policy increases as knowledge of the full object geometry becomes more crucial to task success.

### G. System Capabilities

**Task performance, generalization, robustness.** We evaluate the overall capabilities of our system via task success rates of the best-performing task policy. For each task, we conduct 10 trials for each test object and report the average success rate across all objects. We report a 62.3% success rate for the `grasp-and-reach` task, 80% for the `box lift` task, and 52.5% for the `bimanual handover` task. We test our `grasp-and-reach` policy's ability to generalize to out-of-distribution objects and report qualitative results of successful zero-shot generalization in our video. We also show the robustness against force perturbations of our learned policies for all tasks in Figure 6 and video. More details on the object set for each task are reported in Figure 1.

**Extension to a more capable system.** The learned RL policies can be flexibly chained with finite state machines, teleoperation, etc., to perform longer-horizon tasks while maintaining dexterity, robustness, and generalization. As an example, we present in our video a general pick-and-drop system that can be scripted by utilizing the `grasp-and-reach` policy.

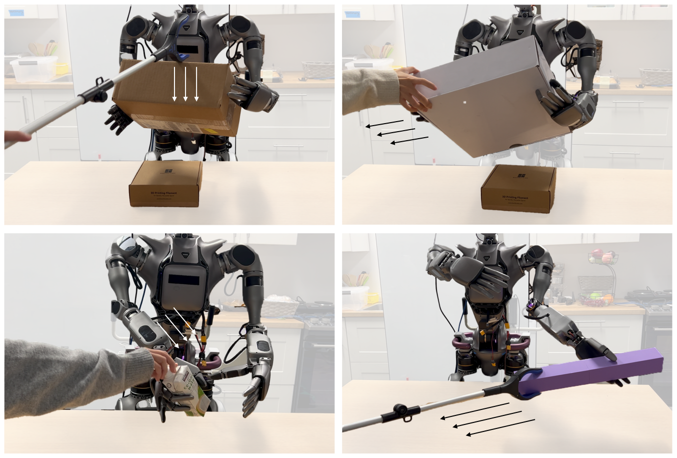
Fig. 6: **Policy robustness.** Our learned policies remain robust under different force perturbations, including knock (top left), pull (top right), push (bottom left), and drag (bottom right).

## V. Limitations

In this work, we investigate the key challenges in applying RL to robot manipulation and introduce practical and principled techniques to overcome the hurdles. Based on the techniques proposed, we build a sim-to-real RL pipeline that demonstrates a feasible path to solve robot manipulation, with evidence on generalizability, robustness, and dexterity.

However, the capabilities achieved in this work are still far from the kind of "general-purpose" manipulation that humans are capable of. Much work remains to be done to improve each individual component of this pipeline and unlock the full potential of sim-to-real RL. For example, the reward design could be improved by integrating even stronger human priors, such as task demonstrations collected from teleoperation.

There are also important open problems that our work does not address. For example, our work uses no novel technique to reduce the sim-to-real gap in dynamics other than applying naive domain randomization. We hypothesize that this could be a reason for the low success rate on `bimanual handover` task, which is the most dynamic among our collection of tasks.

Lastly, we find ourselves heavily constrained by the lack of reliable hardware for dexterous manipulation. While we use multi-fingered robot hands, the dexterity of these hands is far from that of human hands in terms of the active degrees of freedom. We believe the dexterity of our learned policies is not limited by the approach, and we hope to extend our framework to robot hands with more sophisticated designs in the future.

## VI. Conclusion

We present a comprehensive recipe for applying sim-to-real RL to vision-based dexterous manipulation on humanoids. By addressing key challenges in environment modeling, reward design, policy learning, and sim-to-real transfer, we show that RL can be a powerful tool for learning highly useful manipulation skills without the need for extensive human demonstrations. Our learned policies exhibit strong generalization to unseen objects, robustness against force disturbances, and the ability to perform long-horizon contact-rich tasks.

## Acknowledgments

We thank members of NVIDIA GEAR lab for help with hardware infrastructure, in particular Zhenjia Xu, Yizhou Zhao, and Zu Wang. This work was partially conducted during TL's internship at NVIDIA. TL is supported by NVIDIA and the National Science Foundation fellowship.

## References

[1] Open AI, Sep 2024. URL https://openai.com/index/learning-to-reason-with-llms.

[2] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[3] Maria Bauza, Jose Enrique Chen, Valentin Dalibard, Nimrod Gileadi, Roland Hafner, Murilo F Martins, Joss Moore, Rugile Pevceviciute, Antoine Laurens, Dushyant Rao, et al. Demostart: Demonstration-led auto-curriculum applied to sim-to-real with multi-fingered robots. *arXiv preprint arXiv:2409.06613*, 2024.

[4] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

[5] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[6] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[7] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[8] Yuanpei Chen, Chen Wang, Li Fei-Fei, and C Karen Liu. Sequential dexterity: Chaining dexterous policies for long-horizon manipulation. *arXiv preprint arXiv:2309.00987*, 2023.

[9] Yuanpei Chen, Chen Wang, Yaodong Yang, and C Karen Liu. Object-centric dexterous manipulation from human motion data. *arXiv preprint arXiv:2411.04005*, 2024.

[10] Xuxin Cheng, Jialong Li, Shiqi Yang, Ge Yang, and Xiaolong Wang. Open-television: Teleoperation with immersive active visual feedback. *arXiv preprint arXiv:2407.01512*, 2024.

[11] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *RSS*, 2023.

[12] Daniel Dewey. Reinforcement learning and the reward engineering principle. In *2014 AAAI Spring Symposium Series*, 2014.

[13] Jonas Eschmann. *Reward Function Design in Reinforcement Learning*, pages 25–33. Springer International Publishing, Cham, 2021. ISBN 978-3-030-41188-6. doi: 10.1007/978-3-030-41188-6_3. URL https://doi.org/10.1007/978-3-030-41188-6_3.

[14] DeepSeek-AI et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

[15] Kristen Grauman, Andrew Westbury, Lorenzo Torresani, Kris Kitani, Jitendra Malik, Triantafyllos Afouras, Kumar Ashutosh, Vijay Baiyya, Siddhant Bansal, Bikram Boote, et al. Ego-exo4d: Understanding skilled human activity from first-and third-person perspectives. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19383–19400, 2024.

[16] Tuomas Haarnoja, Ben Moran, Guy Lever, Sandy H Huang, Dhruva Tirumala, Jan Humplik, Markus Wulfmeier, Saran Tunyasuvunakool, Noah Y Siegel, Roland Hafner, et al. Learning agile soccer skills for a bipedal robot with deep reinforcement learning. *Science Robotics*, 9(89):eadi8022, 2024.

[17] Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, et al. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5977–5984. IEEE, 2023.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[19] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[20] Binghao Huang, Yuanpei Chen, Tianyu Wang, Yuzhe Qin, Yaodong Yang, Nikolay Atanasov, and Xiaolong Wang. Dynamic handover: Throw and catch with bimanual hands. *arXiv preprint arXiv:2309.05655*, 2023.

[21] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

[22] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[23] Riashat Islam, Peter Henderson, Maziar Gomrokchi, and Doina Precup. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *arXiv preprint arXiv:1708.04133*, 2017.

[24] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, 2023.

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[26] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5 (47):eabc5986, 2020.

[27] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37(4-5):421–436, 2018.

[28] Xuanlin Li, Tong Zhao, Xinghao Zhu, Jiuguang Wang, Tao Pang, and Kuan Fang. Planning-guided diffusion policy learning for generalizable contact-rich bimanual manipulation. *arXiv preprint arXiv:2412.02676*, 2024.

[29] Fanqi Lin, Yingdong Hu, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation. *arXiv preprint arXiv:2410.18647*, 2024.

[30] Toru Lin and Allan Jabri. Mimex: intrinsic rewards from masked input modeling. *Advances in Neural Information Processing Systems*, 36, 2024.

[31] Toru Lin, Zhao-Heng Yin, Haozhi Qi, Pieter Abbeel, and Jitendra Malik. Twisting lids off with two hands. *arXiv:2403.02338*, 2024.

[32] Toru Lin, Yu Zhang, Qiyang Li, Haozhi Qi, Brent Yi, Sergey Levine, and Jitendra Malik. Learning visuotactile skills with two multifingered hands. *arXiv:2404.16823*, 2024.

[33] Minghuan Liu, Zixuan Chen, Xuxin Cheng, Yandong Ji, Ri-Zhao Qiu, Ruihan Yang, and Xiaolong Wang. Visual whole-body control for legged loco-manipulation. *arXiv preprint arXiv:2403.16967*, 2024.

[34] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[35] Tyler Ga Wei Lum, Martin Matak, Viktor Makoviychuk, Ankur Handa, Arthur Allshire, Tucker Hermans, Nathan D Ratliff, and Karl Van Wyk. Dextrah-g: Pixels-to-action dexterous arm-hand grasping with geometric fabrics. *arXiv preprint arXiv:2407.02274*, 2024.

[36] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[37] Marius Memmel, Andrew Wagenmaker, Chuning Zhu, Patrick Yin, Dieter Fox, and Abhishek Gupta. Asid: Active exploration for system identification in robotic manipulation. *arXiv preprint arXiv:2404.12308*, 2024.

[38] Georg Ostrovski, Marc G Bellemare, Aäron Oord, and Rémi Munos. Count-based exploration with neural density models. In *International conference on machine learning*, pages 2721–2730. PMLR, 2017.

[39] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

[40] Aleksei Petrenko, Arthur Allshire, Gavriel State, Ankur Handa, and Viktor Makoviychuk. Dexpbt: Scaling up dexterous manipulation for hand-arm systems with population based training. *arXiv preprint arXiv:2305.12127*, 2023.

[41] Haozhi Qi, Ashish Kumar, Roberto Calandra, Yi Ma, and Jitendra Malik. In-hand object rotation via rapid motor adaptation. In *Conference on Robot Learning*, pages 1722–1732. PMLR, 2023.

[42] Haozhi Qi, Brent Yi, Sudharshan Suresh, Mike Lambeta, Yi Ma, Roberto Calandra, and Jitendra Malik. General in-hand object rotation with vision and touch. In *Conference on Robot Learning*, pages 2549–2564. PMLR, 2023.

[43] Ilija Radosavovic, Tete Xiao, Bike Zhang, Trevor Darrell, Jitendra Malik, and Koushil Sreenath. Real-world humanoid locomotion with reinforcement learning. *arXiv:2303.03381*, 2023.

[44] Ilija Radosavovic, Tete Xiao, Bike Zhang, Trevor Darrell, Jitendra Malik, and Koushil Sreenath. Real-world humanoid locomotion with reinforcement learning. *Science Robotics*, 9(89):eadi9579, 2024.

[45] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[46] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[47] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[48] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[49] Ritvik Singh, Arthur Allshire, Ankur Handa, Nathan Ratliff, and Karl Van Wyk. Dextrah-rgb: Visuomotor policies to grasp anything with dexterous hands. *arXiv preprint arXiv:2412.01791*, 2024.

[50] Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

[51] Richard S Sutton and Andrew G Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, 1998.

[52] Adrien Ali Taïga, William Fedus, Marlos C Machado, Aaron Courville, and Marc G Bellemare. Benchmarking bonus-based exploration methods on the arcade learning environment. *arXiv preprint arXiv:1908.02388*, 2019.

[53] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

[54] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[55] Marcel Torne, Anthony Simeonov, Zechu Li, April Chan, Tao Chen, Abhishek Gupta, and Pulkit Agrawal. Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation. *Arxiv*, 2024.

[56] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782): 350–354, 2019.

[57] Jun Wang, Ying Yuan, Haichuan Che, Haozhi Qi, Yi Ma, Jitendra Malik, and Xiaolong Wang. Lessons from learning to spin" pens". *arXiv preprint arXiv:2407.18902*, 2024.

[58] Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators, 2023.

[59] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018.

[60] Zhao-Heng Yin, Changhao Wang, Luis Pineda, Francois Hogan, Krishna Bodduluri, Akash Sharma, Patrick Lancaster, Ishita Prasad, Mrinal Kalakrishnan, Jitendra Malik, et al. Dexteritygen: Foundation controller for unprecedented dexterity. *arXiv preprint arXiv:2502.04307*, 2025.

[61] Kevin Zakka, Philipp Wu, Laura Smith, Nimrod Gileadi, Taylor Howell, Xue Bin Peng, Sumeet Singh, Yuval Tassa, Pete Florence, Andy Zeng, et al. Robopianist: Dexterous piano playing with deep reinforcement learning. *arXiv preprint arXiv:2304.04150*, 2023.

[62] Chong Zhang, Wenli Xiao, Tairan He, and Guanya Shi. Wococo: Learning whole-body humanoid control with sequential contacts. *arXiv preprint arXiv:2406.06005*, 2024.

[63] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[64] Tony Z Zhao, Jonathan Tompson, Danny Driess, Pete Florence, Kamyar Ghasemipour, Chelsea Finn, and Ayzaan Wahid. Aloha unleashed: A simple recipe for robot dexterity. *arXiv preprint arXiv:2410.13126*, 2024.

[65] Yuke Zhu, Ziyu Wang, Josh Merel, Andrei Rusu, Tom Erez, Serkan Cabi, Saran Tunyasuvunakool, János Kramár, Raia Hadsell, Nando de Freitas, et al. Reinforcement and imitation learning for diverse visuomotor skills. *arXiv preprint arXiv:1802.09564*, 2018.

## Appendix

### A. Environment Modeling Details

**Modeling underactuated joints.** Since modeling underactuated joint structure is not directly supported, we approximate the relationship between each pair of actuated and underactuated joints by fitting a linear function $q_u = k \cdot q_a + b$, where $q_u$ is the underactuated joint angle and $q_a$ is the actuated joint angle. Note that parameters $k, b$ are included as tunable parameters to search over using our autotune module detailed in Section III-A.

### B. Reward Design Details

We design generalizable rewards based on the principle outlined in Section III-B and list task reward details below.

Both **grasp** and **lift** tasks can be defined with the following goal states: (1) finger contact with the object; (2) the object being lifted up to a goal position. Our reward design can, therefore, follow by combining the contact goal reward and the object goal reward terms:

$$r(s_h, s_o) = r_{contact}(s_h, s_o) + r_{goal}(s_o) \tag{3}$$

where $s_h$ includes fingertip positions, $s_o$ includes object center-of-mass position, and all contact marker positions (if any).

Similarly, the **handover** task can be defined with the following goal states: (1) one hand's finger contact with the object; (2) object being transferred to an intermediate goal position while still in contact with the first hand; (3) the second hand's finger contact with the object; (4) object being transferred to the final goal position. Due to the hand switching, we introduce a stage variable $a \in \{0, 1\}$ and design the reward as follows:

$$\begin{aligned} r(s_h, s_o) = (1 - a) \cdot (r_{contact}(s_{h_A}, s_{o_A}) + r_{goal}(s_{o_A})) \\ + a \cdot (r_{contact}(s_{h_B}, s_{o_B}) + r_{goal}(s_{o_B})) \end{aligned} \tag{4}$$

where $s_{h_A}, s_{h_B}$ denote fingertip positions of the engaged hand at each stage, $s_o$ denote object center-of-mass position and desirable contact marker positions (if any) at each stage. At completion of each stage, we also reward policy with a bonus whose scale increases as stage progresses.

### C. Policy Training Details

**RL implementation.** To learn the specialist policies, the observation space includes object position and robot joint position at each time step, and the action space is robot joint angles. We use Proximal Policy Optimization [47] with asymmetric actor-critic as the RL algorithm. In addition to the policy inputs, we provide the following privilege state inputs to the asymmetric critic: arm joint velocities, hand joint velocities, all fingertip positions, object orientation, object velocity, object angular velocity, object mass randomization scale, object friction randomization scale, and object shape randomization scale. Both the actor and critic networks are 3-layer MLPs with units $(512, 512, 512)$.

**Domain randomization.** Physical randomization includes the randomization of object friction, mass, and scale. We also apply random forces to the object to simulate the physical effects that are not implemented by the simulator. Non-physical randomization models the noise in observation (e.g., joint position measurement and detected object positions) and action. A summary of our randomization attributes and parameters is shown in Table IV.

TABLE IV: Domain Randomization Setup.

| | |
|---|---|
| Object: Mass (kg) | [0.03, 0.1] |
| Object: Friction | [0.5, 1.5] |
| Object: Shape | $\times\mathcal{U}(0.95, 1.05)$ |
| Object: Initial Position (cm) | $+\mathcal{U}(-0.02, 0.02)$ |
| Object: Initial $z$-orientation | $+\mathcal{U}(-0.75, 0.75)$ |
| Hand: Friction | [0.5, 1.5] |
| PD Controller: P Gain | $\times\mathcal{U}(0.8, 1.1)$ |
| PD Controller: D Gain | $\times\mathcal{U}(0.7, 1.2)$ |
| Random Force: Scale | 2.0 |
| Random Force: Probability | 0.2 |
| Random Force: Decay Coeff. and Interval | 0.99 every 0.1s |
| Object Pos Observation: Noise | 0.02 |
| Joint Observation Noise. | $+\mathcal{N}(0, 0.4)$ |
| Action Noise. | $+\mathcal{N}(0, 0.1)$ |
| Frame Lag Probability | 0.1 |
| Action Lag Probability | 0.1 |
| Depth: Camera Pos Noise (cm) | 0.005 |
| Depth: Camera Rot Noise (deg) | 5.0 |
| Depth: Camera Field-of-View (deg) | 5.0 |

### D. Distillation Details

To learn the generalist policy, we reduce the choices of observation inputs to the robot joint states and selective object states, including 3D object position and egocentric depth view, since privileged information is unavailable for sim-to-real transfer. To more efficiently utilize the trajectory data and improve training stability, for each sub-task specialist policy, we evaluate for 5000 steps over 100 environments, saving trajectories filtered by success at episode reset on the hard disk. We then treat the saved data as "demonstrations" and learn a generalist policy for each task with Diffusion Policies [11].

The proprioception and object position states are concatenated and passed through a three-layer network with ELU activation, hidden sizes of $(512, 512, 512)$, and an output feature size of 64. For depth observations, we use the ResNet-18 architecture [18] and replace all the BatchNorm [22] in the network with GroupNorm [59], following [11]. All the encoded features are then concatenated as the input to a diffusion model. We use the same noise schedule (square cosine schedule) and the same number of diffusion steps (100) for training as in [11]. The diffusion output from the model is the normalized 7 DoF absolute desired joint positions of each humanoid arm and the 6 DoF normalized (0 to 1) desired joint positions of each humanoid hand. We use the AdamW optimizer [25, 34] with a learning rate of 0.0001, weight decay of 0.00001, and a batch size of 128. Following [11], we maintain an exponential weighted average of the model weights and use it during evaluation/deployment.