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

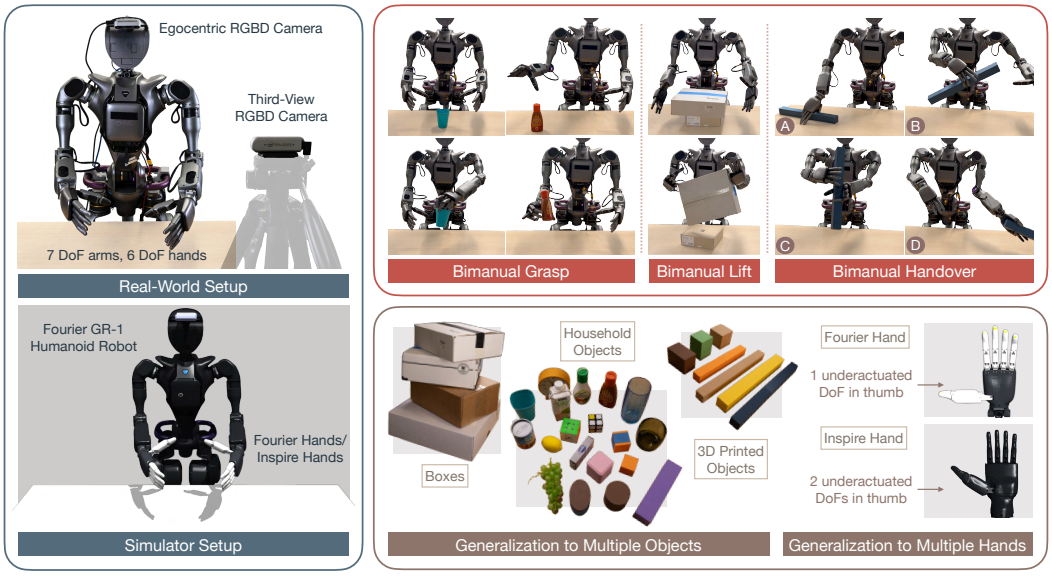

Figure 1: **Overview.** We train a humanoid robot with two multi-fingered hands to perform a range of contact-rich dexterous manipulation tasks on diverse objects. Observations are obtained from a third-view camera, an egocentric camera, and robot proprioception. Our reinforcement learning policies generalize zero-shot to unseen real-world objects with varying physical properties (e.g. shape, size, color, material, mass) and remain robust against force disturbances. We also validate the adaptability of our approach on two hardware variations.

Abstract: Learning generalizable robot manipulation policies, especially for complex multi-fingered humanoids, remains a significant challenge. Existing approaches primarily rely on extensive data collection and imitation learning, which are expensive, labor-intensive, and difficult to scale. Sim-to-real reinforcement learning (RL) offers a promising alternative, but has mostly succeeded in simpler state-based or single-hand setups. How to effectively extend this to vision-based, contact-rich bimanual manipulation tasks remains an open question. In this paper, we introduce a practical sim-to-real RL recipe that trains a humanoid robot to perform three challenging dexterous manipulation tasks: grasp-and-reach, box lift and bimanual handover. Our method features an automated real-to-sim tuning module, a generalized reward formulation based on contact and object goals, a divide-and-conquer policy distillation framework, and a hybrid object representation strategy with modality-specific augmentation. We demonstrate high success rates on unseen objects and robust, adaptive policy behaviors – highlighting that vision-based dexterous manipulation via sim-to-real RL is not only viable, but also scalable and broadly applicable to real-world humanoid manipulation tasks.

Keywords: Humanoids, Vision-Based Dexterous Manipulation, Reinforcement Learning, Sim-to-Real

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

1 Introduction

Learning generalizable manipulation policies – especially for complex humanoid robots equipped with multi-fingered hands – remains a formidable challenge in robotics. Existing approaches often rely on extensive real-world data collection and imitation learning [1, 2, 3], which are costly, labor-intensive, and difficult to scale. Sim-to-real reinforcement learning (RL) offers a promising alternative and has achieved impressive results in navigation [4, 5], locomotion [6, 7], and autonomous drone racing [8]. However, its application to dexterous manipulation remains largely limited to single-hand [9, 10, 11, 12, 13] or state-based setups [14, 15, 16, 17], leaving open the question of how to scale RL to vision-based, contact-rich bimanual tasks on humanoid embodiments.

In this work, we present a practical vision-based sim-to-real RL recipe that enables a multi-fingered humanoid robot to learn highly generalizable, robust, and dexterous manipulation skills. We identify and address several key challenges that have not been thoroughly explored in prior works:

(A) Sim-to-real for low-cost manipulation systems. Existing approaches rely on industry-grade robotic arms with high-precision motors. However, many humanoid platforms employ much more lightweight, noisier motors. This makes contact-rich dexterous grasping and bimanual coordination significantly harder, especially with sim-to-real method. We introduce a simple, automated real-to-sim system identification method to overcome this with less than four minutes of real-world data.
(B) Reward design for complex coordination. Bimanual manipulation tasks such as handover and lifting require complex coordination between arms and hands: one side must act in a way that complements the other, with precision in both motion and timing. Designing a reward function that captures this type of contact-rich collaboration is nontrivial. We propose a novel keypoint-based reward formulation to facilitate such coordination effectively.
(C) Exploration. The long-horizon, high-dimensional nature of bimanual coordination introduces a hard exploration problem, even when reward functions are well-shaped. We propose to use a task-aware initialization strategy to accelerate single task RL, and decompose the overall multi-task (e.g. object) policy learning into separate single-task RL followed by generalist policy distillation.
(D) Object perception The combination of object diversity and sim-to-real domain shift makes vision-based manipulation particularly difficult. We propose a hybrid object representation that combines compact low-dimensional features with expressive high-dimensional features, augmented via modality-specific randomization. We find this simple strategy surprisingly effective – improving sim-to-real success rates on novel objects significantly by 80$\sim$100%.

We demonstrate the effectiveness of our approach on three challenging vision-based manipulation tasks: dexterous grasp-and-reach, bimanual lifting, bimanual handover. Our zero-shot sim-to-real policies exhibit robust, adaptive, and generalizable behavior on unseen real-world objects with diverse physical properties, achieving 90% success rate on seen objects and 60$\sim$80% success rate on novel objects. Additionally, we confirm our method's adaptability to hardware variation across two distinct multi-fingered robot hands. Together, these results establish a practical and scalable recipe for high-performance vision-based dexterous manipulation via sim-to-real RL.

2 Background

2.1 Deep Reinforcement Learning Applications to Robotics

The successes of deep RL across domains like gaming, language modeling, and control [6, 8, 18, 19, 20, 21] have generated widespread excitement. However, the paradigm is known to be brittle, with sensitivity to hyperparameters [22] and reproducibility issues [23] due to high algorithmic variance.

Among open problems in RL, exploration remains fundamental. Unlike supervised learning, RL agents must collect their own data — and the strategy for doing so directly affects performance. Real-world robotics compounds this difficulty with high-dimensional inputs, sparse rewards, and complex dynamics. Numerous methods have aimed to scale exploration by incentivizing novelty [24, 25, 26, 27, 28, 29, 30], but they do not fundamentally resolve the exploration bottleneck.

Robotics also exposes challenges overlooked in standard RL benchmarks [31, 32], including: (1) the absence of fully modeled environments, and (2) the lack of clearly defined reward functions. Past works have introduced practical techniques to mitigate these issues, such as learning from

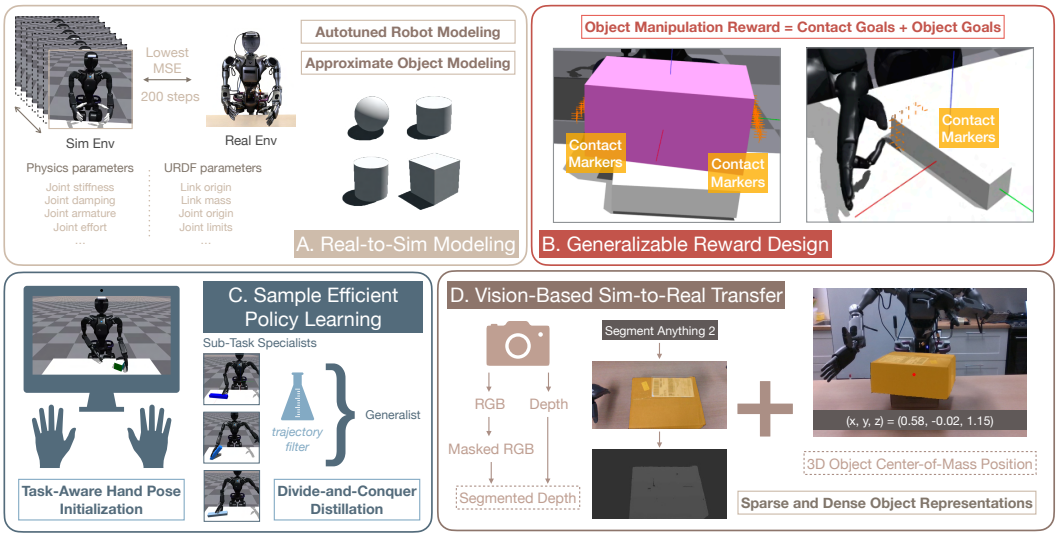

Figure 2: **A sim-to-real RL recipe for vision-based dexterous manipulation.** We close the environment modeling gap between simulation and reality through an automated real-to-sim tuning module, design generalizable task rewards by disentangling each manipulation task into contact states and object states, improve sample efficiency of 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.

motion capture or teleoperated demonstrations [33, 34, 35, 36], real-to-sim modeling techniques [9, 4, 10, 14, 37], and more principled reward design [38, 39]. While often tailored to specific tasks or hardware, these approaches lay groundwork that our method builds upon and generalizes.

2.2 Vision-Based Dexterous Manipulation on Humanoids

Imitation learning and classical approaches. Recent advances in teleoperation [2, 3, 40, 41] and learning from demonstrations [42, 43] have enabled significant progress in vision-based dexterous manipulation [2, 43, 3, 44]. However, teleoperation remains costly to scale, and achieving high success rates with real-world demonstration data alone [45, 46, 44] requires large datasets, making purely supervised methods expensive for reaching human-level performance on complex tasks.

Reinforcement learning approaches. RL-based manipulation works have shown strong results in settings such as in-hand reorientation [9, 10, 12, 47], grasping [17, 13], twisting [14], and dynamic handover [15], but typically focus on single-hand setups [9, 48, 10, 17, 12, 13, 47] or use intermediate object representations rather than raw pixels [33, 15, 14]. The closest to our work is Chen et al. [33], but their method relies on human hand motion capture, while our work learns full hands-arms joint control from scratch. Our work is also the first to demonstrate robust sim-to-real transfer of bimanual policies on a novel humanoid platform with multi-fingered hands.

3 Our Recipe

Section 1 outlined four key challenges in sim-to-real RL for dexterous manipulation. Here, we provide the detailed approaches we develop for each. An overview is shown in Figure 2.

3.1 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 hinges on accurate modeling of both robots and environments. In dexterous manipulation, this challenge is compounded by the necessity to model objects, which have diverse and often unmeasurable physical properties. Even with known parameters, matching real-world and simulated dynamics is nontrivial: the same values for physical constants in simulation and the real world do not necessarily correspond to identical kinematic and dynamic relationships due to discrepancies in physics engines.

Autotuned robot modeling. Manufacturer-supplied robot models offer a baseline, but often require significant tuning [9, 49] to be ready for sim-to-real. This tuning is a laborious process as there is no "ground truth" pairing between the real world and the simulated world. We propose an *autotune* module for fast, automated calibration of simulation parameters to match real robot behavior. As shown in Figure 2A and Algorithm 1, our method jointly optimizes simulator physics (e.g. friction, damping) and URDF constants (e.g. link inertia values, joint limits) using only a single set of calibration trajectories on real hardware. It samples parameter sets, runs joint-targeted motions in parallel simulations, and selects the set minimizing tracking error against the real robot – 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). This removes the need for iterative manual tuning and generalizes to any simulator-exposed parameter affecting kinematic behaviors.

Algorithm 1 Real-to-Sim Autotune Module

Require:
E : Set of environment parameters to tune
N : Number of calibration action sequences
R : Real robot hardware environment
M : Initial robot model file
procedure AUTOTUNE(E, N, R, M)
  $P \leftarrow$ InitializeParameterSpace(E, M)
  $S \leftarrow \{\}$ ▷ Set of simulated environments
  for $i \leftarrow 1$ to K **do** ▷ K is population size
    $p_i \leftarrow$ RandomSample(P)
    $S_i \leftarrow$ CreateSimEnvironment(p_i)
    $S \leftarrow S \cup \{S_i\}$
  end for
  $J \leftarrow$ GenerateJointTargets(N)
  $R_{track} \leftarrow$ GetTrackingErrors(R, J) ▷ Real tracking
  $best\_params \leftarrow$ null
  $min\_error \leftarrow \infty$
  for $S_i \in S$ **do**
    $S_{track} \leftarrow$ GetTrackingErrors(S_i, J)
    $error \leftarrow$ ComputeMSE(S_{track}, R_{track})
    if $error < min\_error$ **then**
      $min\_error \leftarrow error$
      $best\_params \leftarrow$ GetParameters(S_i)
    end if
  end for
  return $best\_params$
end procedure

Approximate object modeling. Following prior work [14, 50], we model objects using simple geometric primitives (e.g. cylinders) with randomized physical parameters. Despite their simplicity, these approximations are sufficient to learn dexterous manipulation policies that transfer reliably to the real world. Our recipe adopts this strategy and finds it both effective and generalizable.

3.2 Generalizable Reward Design

In standard RL [51], the reward function plays a central role in shaping agent behavior. However, much of RL research has treated rewards as fixed, focusing instead on algorithmic improvements [52]. In robotics — and especially in dexterous manipulation — designing effective, generalizable rewards becomes a key challenge due to complex contact dynamics and object variability [53].

Manipulation as contact and object goals. We observe that many human manipulation tasks [54] can be decomposed into a sequence of hand-object contact transitions and object state changes. Inspired by this, we propose a structured reward design scheme for long-horizon, contact-rich tasks. For instance, a bimanual handover can be segmented into: (1) one hand contacting the object, (2) lifting the object near the second hand, (3) the second hand contacting the object, and (4) transferring the object to the target location. We therefore define rewards based on two key components: "contact goals" encourages the fingertips to reach task-relevant contact points on object, and "object goals" penalizes current object state deviation from the target object state (e.g. *xyz* position). To facilitate contact goal specification, we introduce a keypoint-based technique: simulated objects are augmented with "contact stickers" — surface markers representing desirable contact locations. The contact goal, in terms of reward, can then be specified as $r_{\text{contact}} = \sum_i \left[\frac{1}{1+\alpha d(\mathbf{X}^L, \mathbf{F}_i^L)} + \frac{1}{1+\beta d(\mathbf{X}^R, \mathbf{F}_i^R)} \right]$, 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, α and β are scaling hyperparameters, and d is a distance function defined as $d(\mathbf{A}, \mathbf{x}) = \min_i \|\mathbf{A}_i - \mathbf{x}\|_2$. These contact markers can be arbitrarily specified – for example, procedurally generated based on object geometry – offering a flexible way to incorporate contact preferences or human priors. A visualization of contact markers is shown in Figure 2B, and their empirical effectiveness is analyzed in Section 4.

3.3 Sample Efficient Policy Learning

Even with a well-shaped reward, learning dexterous policies on high-dimensional bimanual multi-fingered systems remains sample-inefficient due to sparse rewards and exploration complexity. We introduce two techniques to improve sample efficiency: (1) task-aware initialization using human-guided hand poses, and (2) a divide-and-conquer strategy with policy distillation.

Task-aware hand poses for initialization. We collect task-relevant hand-object configurations from human teleoperation in simulation. This can be done using any compatible system for bimanual multi-fingered hands. The recorded states, including object poses and robot joint positions, are then randomly sampled as initial conditions for each training episode. Unlike prior work that relies on full demonstration trajectories [55], our approach only requires humans to casually "play around" with the task goal in mind. This lightweight data collection takes less than 30 seconds per task since no expert demonstration is needed, yet proves highly effective in improving early-stage exploration.

Divide-and-conquer distillation. Standard RL exploration techniques [25, 26, 28, 56] aim to visit the state space more efficiently but do not fundamentally alter the difficulty of sparse-reward problems: the probability of receiving learning signals from visiting the "right" states remains the same. We instead overcome the exploration problem by breaking down the explorable state space itself, e.g. decomposing a multi-object manipulation task into multiple single-object manipulation tasks. Once specialized policies are trained for each sub-task, high-quality rollouts can be filtered and distilled into a generalist policy using shared observation and action spaces. This effectively brings pure RL closer to learning from demonstrations, where the sub-task policies act as "teleoperators" in the simulation environment, and the centralized generalist policy learns from curated data.

3.4 Vision-Based Sim-to-Real Transfer

Vision-based sim-to-real transfer is particularly challenging due to domain gaps in both dynamics and perception. We employ two strategies to address these challenges: hybrid object representations and extensive domain randomization.

Hybrid object representations. Dexterous manipulation often requires precise perception of object pose and geometry. Prior work spans a spectrum of object representations, from 3D position [14] and 6D pose [9], to depth [17, 12], point cloud [57], and RGB images [10]. Higher-dimensional representations encode richer information about the object, improving task performance but also widening the sim-to-real gap; and vice versa. To balance the trade-offs, we propose to use a mix of low- and high-dimensional signals: 3D object position (from a fixed third-person view) and depth image (from an egocentric view). We obtain the 3D object position from a reliable object tracking module with relatively controllable noise, and use segment out the object depth to reduce the visual sim-to-real gap. We validate this design in Section 4.

Domain randomization for perception and dynamics. To improve robustness, we apply extensive domain randomization during training. This includes variation in object parameters, camera parameters, robot physical properties, and observation noises. Full details are provided in Appendix 6.3.

4 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 our website.

4.1 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 6 actuated DoFs and 5 underactuated DoFs. To show cross-embodiment generalization, we include results on the Inspire hands, each with 6 actuated DoFs and 6 underactuated DoFs. The hardware has substantially different masses, surface frictions, finger and palm morphologies, and thumb actuations. Figure 1 visualizes both hands. We use the NVIDIA Isaac Gym simulator [58].

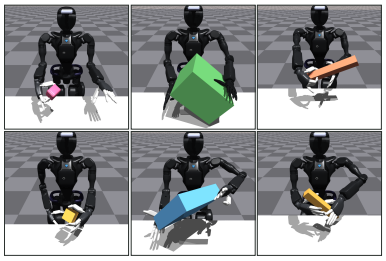

Figure 3: **Policies learned in simulation.** Left: grasp-and-reach; middle: box lift; right: bimanual handover (right-to-left, left-to-right).

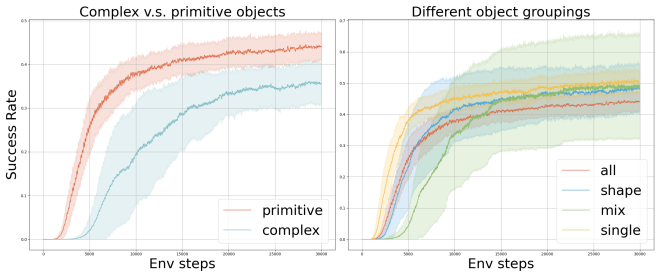

Figure 4: **Training `grasp-and-reach` policy with different object sets.** Each curve is from 10 runs with different random seeds. Left: training with complex objects v.s. simple geometric primitive objects. Right: training with differently grouped geometric objects.

Perception. As outlined in Section 3.4, 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 RealSense D435 depth camera on the head of the humanoid robot and a third-view RealSense D435 depth camera on a tripod in front of the robot (illustrated in Figure 1). In simulation, we similarly set up the two cameras by calibrating their 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. [14]. As illustrated in Figure 2, we use the Segment Anything Model 2 (SAM2) [59] to generate a segmentation mask for the object at each trajectory sequence's initial frame, and leverage the tracking capabilities of SAM2 to track the mask throughout all remaining frames. To approximate object's 3D center-of-mass coordinates, we calculate the center position of object mask 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.

4.2 Task Definition

(A) Grasp-and-reach. The robot must use one hand to grasp a tabletop object, lift it, and place it at a goal location. At initialization, a scripted vision module selects the closer hand to object. Test objects vary in shape, mass, volume, friction, color, and texture (see Figure 1). Each trial randomizes object pose and goal location. **(B) Box lift.** The robot lifts a box too large for single-handed grasping. Box size, color, mass, and initial pose (with randomized position and yaw) are varied across trials. **(C) Bimanual handover.** The robot grasps an object from one side of the table with one hand and hands it over to the other hand, which cannot reach the object directly. Objects vary in color, size, mass, and pose. We vary the initial pose of blocks in each trial.

4.3 Evaluation of Real-to-Sim Modeling

Effectiveness of autotuned robot modeling. We apply the autotune module described in Section 3.1 to optimize the robot modeling parameters. To assess its effectiveness, we compare the sim-to-real transfer success rates of three sets of policy checkpoints, each trained with identical settings except for the robot modeling parameters. These parameter sets correspond to varying levels of modeling accuracy, as measured by the mean squared error (MSE) from autotune – ranging from the lowest (i.e., smallest real-to-sim gap) to the highest (i.e., largest real-to-sim gap). As shown in Table 1, policies trained with autotuned models exhibit significantly better sim-to-real performance. Qualitative examples in our video further demonstrate successful transfer of `grasp-and-reach` policies to the Inspire hands, highlighting the generalizability of our autotune module.

Effectiveness of approximate object modeling. Empirically, 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. As shown in Figure 4 (left), training `grasp-and-reach` policies with primitive shapes leads to faster convergence compared to using complex object ge-

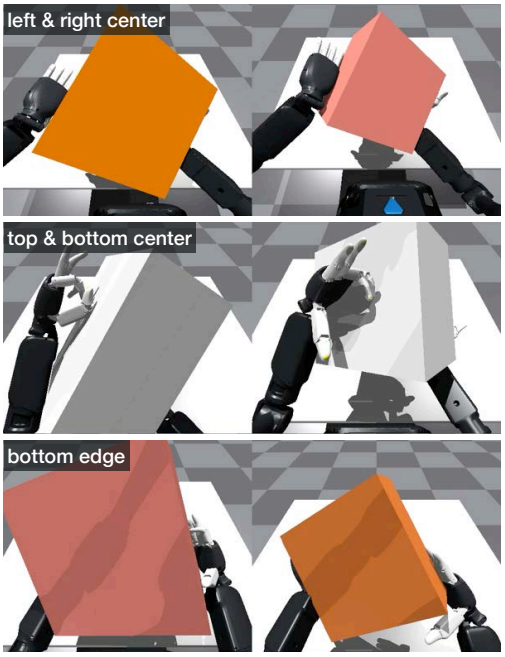

Figure 5: **Different contact patterns emerge from different placements of contact markers.** Top: contact markers on the left and right side centers; middle: markers on the top and bottom side centers; bottom: markers on the bottom side edges.

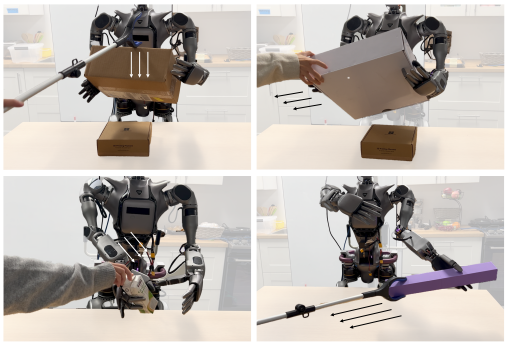

Figure 6: **Policy robustness.** Our learned policies remain robust under different force perturbations, including knock (top left), pull (top right), push (bottom left), and drag (bottom right).

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

Table 1: **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.

ometries. More importantly, policies trained with randomized primitive shapes demonstrate strong generalization to a diverse set of unseen objects, as illustrated in our video.

4.4 Evaluation of Reward Design

Task capabilities. Enabled by our proposed reward design principle, a broad range of long-horizon, contact-rich tasks can be successfully solved using pure RL, as shown in Figure 3 and video. The resulting policies exhibit notable dexterity and robustness under various random force disturbances.

Effectiveness of contact-based rewards. In Figure 5, we visualize how different contact behaviors emerge from varying the placement of contact markers, using the `box lift` task as an example. The contact stickers are procedurally generated along box sides or edges based on the box dimensions. The resulting behaviors closely reflect the specified contact positions, demonstrating the effectiveness of using contact markers to define contact goals.

4.5 Evaluation of Policy Learning

Effectiveness of task-aware hand pose initialization. In Table 2, we compare the percentage of successfully trained policies for each task with and without task-aware hand pose initialization. The results indicate that incorporating human priors at initialization significantly enhances exploration efficiency in challenging RL tasks.

Divide-and-conquer distillation. We evaluate our divide-and-conquer distillation strategy through two ablation studies. First, we examine how the granularity of task decomposition affects training efficiency in a multi-object `grasp-and-reach` task involving 10 objects. We compare four designs: (1) training a single policy on all objects (*all*); (2) training three policies on shape-similar object groups (*shape*); (3) training three policies on shape-diverse object groups (*mix*); and (4) training ten separate single-object policies (*single*). As shown in Figure 4, *single* achieves the highest sample efficiency, followed by *shape*, *all*, and *mix*. The average success rates also vary across designs, reflecting differences in task difficulty. Notably, policies trained on reduced object sets

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

Table 2: **Initializing with human data.** Correlation between the percentage of successful task policies and whether human play data is used for initialization. We define *successful* 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 3: **Comparing 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. Pickup success measures how often hands pick up the object. Combining 3D position with depth enables easier sim-to-real transfer.

converge to similar final performance, while the *all* policy consistently underperforms. Second, we evaluate the sim-to-real transfer success rate of each policy type on an in-distribution object over 30 trials. The *mix* policy performs best (90.0%), followed by *shape* (63.3%), *single* (40.0%), and *all* (23.3%). We hypothesize that the lower performance of *single* and *mix* arises from overfitting to specific geometries, while the poor performance of *all* is consistent with its weaker RL training outcomes. These results suggest that divide-and-conquer distillation strikes a favorable balance between training efficiency and sim-to-real generalization.

4.6 Evaluation of Vision-Based Sim-to-Real Transfer

Effectiveness of mixing object representations. We study the impact of different object representations on sim-to-real transfer performance, with results summarized in Table 3. Our findings show that combining a dense representation (segmented depth image) with a sparse representation (3D object center-of-mass position) leads to improved transfer success. Notably, the performance gap between the combined depth-and-position policy and the depth-only policy widens for tasks where accurate understanding of full object geometry is more critical to success.

4.7 System Capabilities

Task performance, generalization, robustness. We evaluate the overall effectiveness of our system by reporting task success rates using the best-performing policy for each task. For each task, we perform 10 trials for each test object and compute the average success rate across all objects. We report a 62.3% success rate for the `grasp-and-reach` task, 80% for `box lift`, and 52.5% for `bimanual handover`. To assess generalization, we test the `grasp-and-reach` policy on out-of-distribution objects and present qualitative evidence of successful zero-shot transfer in our video. Additionally, we evaluate the robustness of our policies under external force perturbations across all tasks, as shown in Figure 6 and our videos. 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 seamlessly integrated with higher-level control structures such as finite state machines or teleoperation frameworks to enable longer-horizon task execution, while preserving dexterity, robustness, and generalization. As a proof of concept, our video showcases a general pick-and-drop system constructed by scripting sequences around the `grasp-and-reach` policy.

5 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.

6 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; alternative controller, such as torque controller, could also be explored.

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.

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] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[2] X. Cheng, J. Li, S. Yang, G. Yang, and X. Wang. Open-television: Teleoperation with immersive active visual feedback. *arXiv preprint arXiv:2407.01512*, 2024.

[3] T. Lin, Y. Zhang, Q. Li, H. Qi, B. Yi, S. Levine, and J. Malik. Learning visuotactile skills with two multifingered hands. *arXiv:2404.16823*, 2024.

[4] T. Haarnoja, B. Moran, G. Lever, S. H. Huang, D. Tirumala, J. Humplik, M. Wulfmeier, S. Tunyasuvunakool, N. Y. Siegel, R. Hafner, et al. Learning agile soccer skills for a bipedal robot with deep reinforcement learning. *Science Robotics*, 9(89):eadi8022, 2024.

[5] N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter. Advanced skills by learning locomotion and local navigation end-to-end. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2497–2503. IEEE, 2022.

[6] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

[7] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[8] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, 2023.

[9] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[10] A. Handa, A. Allshire, V. Makoviychuk, A. Petrenko, R. Singh, J. Liu, D. Makoviichuk, K. Van Wyk, A. Zhurkevich, B. 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.

[11] T. Chen, J. Xu, and P. Agrawal. A system for general in-hand object re-orientation. In *Conference on Robot Learning*, pages 297–307. PMLR, 2022.

[12] H. Qi, B. Yi, S. Suresh, M. Lambeta, Y. Ma, R. Calandra, and J. Malik. General in-hand object rotation with vision and touch. In *Conference on Robot Learning*, pages 2549–2564. PMLR, 2023.

[13] R. Singh, A. Allshire, A. Handa, N. Ratliff, and K. Van Wyk. Dextrah-rgb: Visuomotor policies to grasp anything with dexterous hands. *arXiv preprint arXiv:2412.01791*, 2024.

[14] T. Lin, Z.-H. Yin, H. Qi, P. Abbeel, and J. Malik. Twisting lids off with two hands. *arXiv:2403.02338*, 2024.

[15] B. Huang, Y. Chen, T. Wang, Y. Qin, Y. Yang, N. Atanasov, and X. Wang. Dynamic handover: Throw and catch with bimanual hands. *arXiv preprint arXiv:2309.05655*, 2023.

[16] H. Qi, A. Kumar, R. Calandra, Y. Ma, and J. Malik. In-Hand Object Rotation via Rapid Motor Adaptation. In *Conference on Robot Learning (CoRL)*, 2022.

[17] T. G. W. Lum, M. Matak, V. Makoviychuk, A. Handa, A. Allshire, T. Hermans, N. D. Ratliff, and K. Van Wyk. Dextrah-g: Pixels-to-action dexterous arm-hand grasping with geometric fabrics. *arXiv preprint arXiv:2407.02274*, 2024.

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

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

[20] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[21] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354, 2019.

[22] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[23] R. Islam, P. Henderson, M. Gomrokchi, and D. Precup. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *arXiv preprint arXiv:1708.04133*, 2017.

[24] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

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

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

[27] G. Ostrovski, M. G. Bellemare, A. Oord, and R. Munos. Count-based exploration with neural density models. In *International conference on machine learning*, pages 2721–2730. PMLR, 2017.

[28] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

[29] B. C. Stadie, S. Levine, and P. Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

[30] H. Tang, R. Houthooft, D. Foote, A. Stooke, O. Xi Chen, Y. Duan, J. Schulman, F. DeTurck, and P. Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

[31] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279, 2013.

[32] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[33] Y. Chen, C. Wang, Y. Yang, and C. K. Liu. Object-centric dexterous manipulation from human motion data. *arXiv preprint arXiv:2411.04005*, 2024.

[34] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[35] Z.-H. Yin, C. Wang, L. Pineda, F. Hogan, K. Bodduluri, A. Sharma, P. Lancaster, I. Prasad, M. Kalakrishnan, J. Malik, et al. Dexteritygen: Foundation controller for unprecedented dexterity. *arXiv preprint arXiv:2502.04307*, 2025.

[36] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, et al. Reinforcement and imitation learning for diverse visuomotor skills. *arXiv preprint arXiv:1802.09564*, 2018.

[37] M. Torne, A. Simeonov, Z. Li, A. Chan, T. Chen, A. Gupta, and P. Agrawal. Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation. *Arxiv*, 2024.

[38] M. Memmel, A. Wagenmaker, C. Zhu, P. Yin, D. Fox, and A. Gupta. Asid: Active exploration for system identification in robotic manipulation. *arXiv preprint arXiv:2404.12308*, 2024.

[39] C. Zhang, W. Xiao, T. He, and G. Shi. Wococo: Learning whole-body humanoid control with sequential contacts. *arXiv preprint arXiv:2406.06005*, 2024.

[40] P. Wu, Y. Shentu, Z. Yi, X. Lin, and P. Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators, 2023.

[41] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[42] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *RSS*, 2023.

[43] X. Li, T. Zhao, X. Zhu, J. Wang, T. Pang, and K. Fang. Planning-guided diffusion policy learning for generalizable contact-rich bimanual manipulation. *arXiv preprint arXiv:2412.02676*, 2024.

[44] T. Z. Zhao, J. Tompson, D. Driess, P. Florence, K. Ghasemipour, C. Finn, and A. Wahid. Aloha unleashed: A simple recipe for robot dexterity. *arXiv preprint arXiv:2410.13126*, 2024.

[45] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. 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.

[46] F. Lin, Y. Hu, P. Sheng, C. Wen, J. You, and Y. Gao. Data scaling laws in imitation learning for robotic manipulation. *arXiv preprint arXiv:2410.18647*, 2024.

[47] J. Wang, Y. Yuan, H. Che, H. Qi, Y. Ma, J. Malik, and X. Wang. Lessons from learning to spin" pens". *arXiv preprint arXiv:2407.18902*, 2024.

[48] Y. Chen, C. Wang, L. Fei-Fei, and C. K. Liu. Sequential dexterity: Chaining dexterous policies for long-horizon manipulation. *arXiv preprint arXiv:2309.00987*, 2023.

[49] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath. Real-world humanoid locomotion with reinforcement learning. *Science Robotics*, 9(89):eadi9579, 2024.

[50] H. Qi, A. Kumar, R. Calandra, Y. Ma, and J. Malik. In-hand object rotation via rapid motor adaptation. In *Conference on Robot Learning*, pages 1722–1732. PMLR, 2023.

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

[52] J. 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.

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

[54] K. Grauman, A. Westbury, L. Torresani, K. Kitani, J. Malik, T. Afouras, K. Ashutosh, V. Baiyya, S. Bansal, B. 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.

[55] M. Bauza, J. E. Chen, V. Dalibard, N. Gileadi, R. Hafner, M. F. Martins, J. Moore, R. Pevceviciute, A. Laurens, D. Rao, et al. Demostart: Demonstration-led auto-curriculum applied to sim-to-real with multi-fingered robots. *arXiv preprint arXiv:2409.06613*, 2024.

[56] A. A. Taïga, W. Fedus, M. C. Machado, A. Courville, and M. G. Bellemare. Benchmarking bonus-based exploration methods on the arcade learning environment. *arXiv preprint arXiv:1908.02388*, 2019.

[57] M. Liu, Z. Chen, X. Cheng, Y. Ji, R.-Z. Qiu, R. Yang, and X. Wang. Visual whole-body control for legged loco-manipulation. *arXiv preprint arXiv:2403.16967*, 2024.

[58] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[59] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[60] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[61] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[62] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[63] Y. Wu and K. He. Group normalization. In *ECCV*, 2018.

[64] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[65] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Appendix

6.1 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 3.1.

6.2 Reward Design Details

We design generalizable rewards based on the principle outlined in Section 3.2 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:

$$r(s_h, s_o) = r_{contact}(s_h, s_o) + r_{goal}(s_o) \tag{1}$$

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:

$$\begin{aligned} r(s_h, s_o) = (1 - a) \cdot (r_{contact}(s_{h_A}, s_{o_A}) + r_{goal}(s_{o_A})) \\ + a \cdot (r_{contact}(s_{h_B}, s_{o_B}) + r_{goal}(s_{o_B})) \end{aligned} \tag{2}$$

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.

6.3 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 [60] 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 4.

6.4 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

Table 4: 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

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 [42].

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 [61] and replace all the BatchNorm [62] in the network with GroupNorm [63], following [42]. 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 [42]. 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 [64, 65] with a learning rate of 0.0001, weight decay of 0.00001, and a batch size of 128. Following [42], we maintain an exponential weighted average of the model weights and use it during evaluation/deployment.