LodeStar: Long-horizon Dexterity via Synthetic Data Augmentation from Human Demonstrations

Weikang Wan[1]* **Jiawei Fu**[1]* **Xiaodi Yuan**[1] **Yifeng Zhu**[2] **Hao Su**[1]
[1]University of California San Diego [2]The University of Texas at Austin

Abstract: Developing robotic systems capable of robustly executing long-horizon manipulation tasks with human-level dexterity is challenging, as such tasks require both physical dexterity and seamless sequencing of manipulation skills while robustly handling environment variations. While imitation learning offers a promising approach, acquiring comprehensive datasets is resource-intensive. In this work, we propose a learning framework and system LODESTAR that automatically decomposes task demonstrations into semantically meaningful skills using off-the-shelf foundation models, and generates diverse synthetic demonstration datasets from a few human demos through reinforcement learning. These sim-augmented datasets enable robust skill training, with a Skill Routing Transformer policy effectively chaining the learned skills together to execute complex long-horizon manipulation tasks. Experimental evaluations on three challenging real-world long-horizon dexterous manipulation tasks demonstrate that our approach significantly improves task performance and robustness compared to previous baselines. Videos are available at lodestar-robot.github.io.

Keywords: Dexterous Manipulation, Imitation Learning, Sim-to-Real

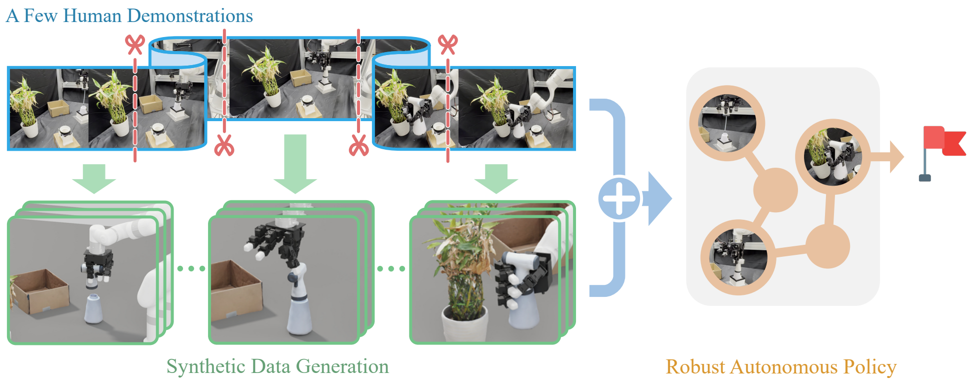

Figure 1: **We present LODESTAR, a framework for learning robust long-horizon dexterous manipulation from a few human demonstrations.** LODESTAR segments demonstrations into sequential skills using off-the-shelf foundation models, and augments each skill via simulation-based residual reinforcement learning. The synthesized skill data is co-trained with real-world data, and a Skill Routing Transformer composes the learned skills into a robust autonomous policy capable of completing complex real-world tasks.

1 Introduction

Developing multi-fingered robotic systems capable of long-horizon manipulation with human-level dexterity has been a longstanding goal in robotics research. Consider everyday activities such as watering a plant with a spray bottle as shown in Fig. 1: this requires not only physical dexterity to

*Equal contribution.

insert and twist the nozzle, lift and hold the bottle upright, and press the trigger, but also the ability to seamlessly sequence these distinct behaviors while robustly handling environment variations. Imitation learning (IL) from human demonstrations [1–3] offers an effective pathway for teaching robots such behaviors. However, for such tasks, acquiring massive, comprehensive, and diverse datasets necessary to ensure policy robustness across potential perturbations and environmental variations encountered in the real world is often costly and resource-intensive [4, 5]. This challenge motivates our investigation into methods that leverage structured data generation from limited demonstrations, aiming to enable robots to perform long-horizon dexterous manipulation tasks robustly.

Prior works have explored generating large datasets from a few demonstrations using replay-based transformation with motion planning [6–9]. While effective, these methods are often limited to parallel grippers or fixed-hand motions. Moreover, naive composing the adapted segments can lead to low-quality or low-diversity demonstrations, hindering generalizable policy learning. Another direction leverages reinforcement learning (RL) in simulation to synthesize diverse data. Recent advances have enabled dexterous skills such as grasping [10–12] and in-hand reorientation [13–15], as RL exploration can discover varied and robust behaviors. However, these results are typically limited to single-stage tasks, rely on hand-tuned simulators and rewards, and remain sensitive to the sim-to-real gap. In parallel, several works decompose long-horizon manipulation into sequential skills and chain them to solve complex tasks [16–21], reducing both the complexity of data generation and policy learning. Yet, smooth and reliable skill transfer remains challenging due to state distribution mismatches at skill boundaries [22], often requiring additional regularization [23, 24] or optimization [25] techniques, which may destabilize policy learning. Despite these advances, robust long-horizon dexterous manipulation from limited data still faces two key challenges: (1) how to automatically segment tasks into semantically meaningful skills with appropriate initial and terminal state distributions; and (2) how to generate diverse training data to robustify individual skills, and effectively chain them into a coherent policy that completes the full task.

To address these challenges, we propose a structured and scalable framework LODESTAR that decomposes long-horizon dexterous manipulation into three stages: skill segmentation, synthetic data generation for robust skill learning, and skill composition via a *Skill Routing Transformer (SRT)* policy. We first segment demonstrations into sequential manipulation skills and intermediate transition motions by leveraging foundation models (VFMs, VLMs) [26–28] to extract visual, spatial, and contact cues directly from raw videos, avoiding manual annotation or predefined primitive skills. For each manipulation skill, we train a robust policy via residual reinforcement learning in simulation environments generated through real-to-sim techniques with augmentation and domain randomization. The learning process is grounded in a small number of real demonstrations, ensuring task relevance and avoiding reward engineering. We further synthesize diverse and physically feasible transition trajectories between skills, eliminating the need for explicit goal specification or slow execution speed of real-world motion planning. Finally, we train a SRT policy on the generated data to compose the learned skills, enabling coherent execution of the full long-horizon task. Our key contribution is a scalable learning framework LODESTAR for robust long-horizon dexterous manipulation from a few human demos. By combining automatic segmentation via foundation models, real-to-sim synthetic data generation for skill acquisition, and skill composition for sim-to-real deployment, LODESTAR enables complex manipulation sequences without requiring extensive task-specific data or manual engineering effort. We validate its effectiveness on three challenging real-world tasks, showing that it outperforms SOTA baselines by an average of 25% success rate.

2 Related Work

2.1 Dexterous Manipulation

Dexterous manipulation remains a long-standing and significant challenge in robotics [29–32], primarily due to the high degrees of freedom involved. Traditional planning and control approaches [33–36] often rely on simplified system models, which struggle to scale to complex, long-horizon tasks. Recent advances have demonstrated the effectiveness of learning-based methods for specific dexterous skills such as grasping [10–12, 37], in-hand reorientation [14, 15, 38], and tool use [39–43]. However, these methods typically target isolated skills rather than integrated, sequential behaviors. In contrast, we tackle the more challenging task of sequencing multiple dexterous skills [25], which requires both fine-grained control and temporal coordination.

2.2 Synthetic Data Generation for Robotic Manipulation

Synthetic simulation data have been used to alleviate the burden of real-world data collection and improve policy generalization in robotic manipulation. Recent studies [9, 44–52] show that co-training policy with simulation data can greatly improve policy performance and robustness by diversifying the training distribution beyond what real-world data alone can offer. Real-to-sim approaches have been used to generate realistic and diverse simulation assets and scenes from real-world inputs via 3D reconstruction [53–59], inverse graphics [60], and foundation model-assisted generation [45, 61–63]. Techniques for sim-to-real transfer include domain randomization [13, 38, 46, 64–67], system identification [68–73] and simulator augmentation [44, 74, 75]. However, naive randomization and augmentation are insufficient to close the sim-to-real gap [46, 62], particularly in long-horizon tasks involving multi-stage and multi-object interactions, where the large variation space hinders effective learning. To address this, our pipeline improves training robustness and efficiency by segmenting demonstrations into multiple stages and applying demonstration-guided augmentation and randomization to expand simulation coverage for each stage.

2.3 Skill Chaining for Long-horizon Robotic Manipulation

Established methods tackle long-horizon tasks by decomposing them into simpler, reusable subtasks, reducing the agent's decision burden via temporally extended actions or skills [76, 77]. Skill discovery approaches fall into three categories: predefined measures [6–8, 22, 78–85], skill discovery from demonstrations [17, 86–94], and unsupervised self-exploration [16, 19, 95–100]. Naively sequential chaining skills often lead to "hand-off" failures, where the terminal state of one skill falls outside the feasible range of the succeeding skill. Prior work mitigates this by updating each skill to encompass the terminal state of the preceding skill [16–21], or by aligning skill boundaries through reward shaping via adversarial [23, 24] or bidirectional training [25]. Closest to our work, [78] learns a transition policy to bridge skills, but assumes it can be trained via random exploration and is limited to simple, predefined primitives. However, these methods typically require either extensive environment interaction [24, 25, 78, 79, 83], annotated demonstrations [6, 8], or high computational resources [101, 102], limiting their scalability to contact-rich, high-DoF dexterous tasks. Our work addresses this by segmenting skills from a small number of demonstrations, and combines insights from skill chaining with recent advances in foundation models [26, 27, 103].

3 Problem Formulation

We formulate our learning problem as a Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, R, P, \rho_0, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $R(s, a, s')$ is the reward function, $P(s' \mid s, a)$ is the transition distribution, ρ_0 is the initial state distribution, and γ is the discount factor. We aim to solve a long-horizon task $\mathcal{T}$, which can potentially be decomposed into a sequence of subtasks $\{\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_K\}$, where K is the total number of subtasks and varies for each long-horizon task. Each subtask $\mathcal{T}_i$ is associated with an initiation set $\mathcal{I}_i \subset \mathcal{S}$ and a termination set $\mathcal{E}_i \subset \mathcal{S}$, indicating the valid start and end states for completing subtask $\mathcal{T}_i$. A policy $\pi : \mathcal{S} \to \mathcal{A}$ induces a distribution over trajectories, and is optimized to maximize the expected sum of discounted rewards $\mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \gamma^t R(s_t, a_t, s_{t+1})\right]$, where T is the episode horizon. We assume access to a small set of full expert demonstrations $\mathcal{D}^e = \{\tau_j^e\}_{j=1}^N$, where τ_j^e is a trajectory of state-action pairs.

4 Approach

We introduce LODESTAR, a framework designed to enable robust long-horizon dexterous manipulation by leveraging a few human demonstrations and sim-augmented data. An overview of the system architecture is shown in Fig. 2. LODESTAR comprises three key components: **(1) Skill Segmentation (Sec. 4.1):** decomposing human demonstrations into motion and manipulation phases using

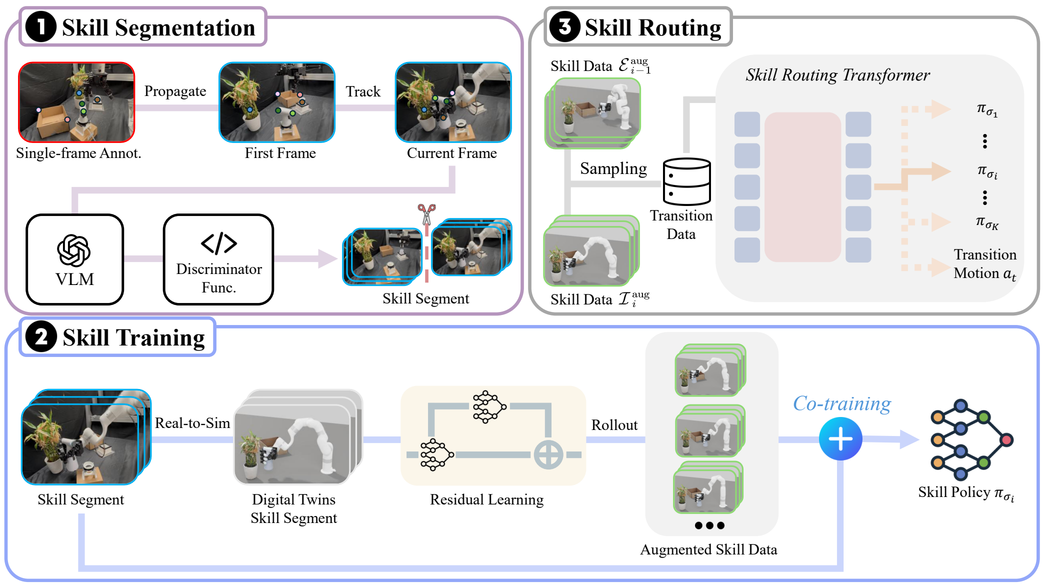

Figure 2: **LODESTAR Pipeline.** LODESTAR consists of three sequential stages. **(1)** Human demonstrations are segmented into manipulation and transition phases using discriminators built from VLM. **(2)** Each skill is augmented via residual RL in simulation to train robust policies.**(3)** Skill Routing Transformer policy composes and selects skills or transition actions during execution, enabling coherent long-horizon manipulation in real-world settings.

per-frame skill discriminators derived from vision-language models and tracked keypoints; **(2) Synthetic Data Generation for Robust Skill Policies Learning (Sec. 4.2):** training individual robust skill policies enhanced by synthetic demonstrations from residual reinforcement learning (RL) policy; **(3) Skill Composition via Skill Routing Transformer (SRT) policy (Sec. 4.3):** learning a SRT policy to sequence and compose the learned skills to accomplish long-horizon tasks.

4.1 Skill Segmentation

In long-horizon dexterous manipulation, human behavior often exhibits a natural decomposition which alternates between two modes: (i) *transition* motions corresponding to inter-skill transit movements [8, 104] that involve either no object interaction or only passive contact, and (ii) *manipulation skill* involving fine-grained, contact-rich manipulations primarily actuated by the hand. We propose decomposing each expert demonstration $\tau_j^e = \{(s_t, a_t)\}_{t=1}^T$ into an alternating sequence of manipulation skill and transition motion segments: $\tau^e = \tau_{\sigma_1} \rightarrow \tau_{\mu_1} \rightarrow \tau_{\sigma_2} \rightarrow \cdots \rightarrow \tau_{\mu_{K-1}} \rightarrow \tau_{\sigma_K}$, where each τ_{σ_i} denotes the demonstration of a semantically meaningful manipulation skill σ_i (e.g., inserting, twisting) to solve the subtask $\mathcal{T}_i$, and each τ_{μ_i} denotes a motion segment serving as a transition between skills. To identify skill segments, we propose a pipeline that leverages vision foundation models and vision-language models to construct a per-frame *discriminator function* for each skill. Each discriminator $d_i(s_t) : \mathcal{S} \rightarrow \{0, 1\}$ determines whether a given frame belongs to the skill σ_i, and is formulated as a conjunction of two constraints: $d_i(s_t) = \mathbb{1}\left[C_i^{\text{point}}(s_t) \wedge C_i^{\text{contact}}(s_t)\right]$, where $C_i^{\text{point}}(\cdot)$ is a keypoint spatial relation constraint and $C_i^{\text{contact}}(\cdot)$ is a fingertip contact constraint which describes whether the fingertip has contact with the object of interest (see Appendix for more details). Unlike prior works [105, 106] that define a skill by its terminal state distribution $\mathcal{E}_i$, we classify skills at the frame level. This formulation is more tractable in dexterous skills, where end states are typically task-specific and diverse across human demonstrations [25], whereas the execution process exhibits consistent spatial and contact patterns.

Keypoint Proposal. To obtain skill-centric spatial constraints, we annotate N semantically meaningful keypoints $\{\hat{p}_l\}_{l=1}^N$ on task-relevant objects in the first frame of a random reference demonstration for each task. These annotated keypoints serve as priors for other demonstrations. We then use the semantic correspondence model DIFT [27] to propagate the initial keypoints to the first frame of other demonstrations, obtaining consistent keypoints across demonstrations. Then, we apply Co-Tracker [26] to track the keypoints $\{p_l^{(j)}(t)\}_{t=1}^{T}$ across each trajectory. This procedure yields per-frame object keypoints, with only a single manual annotation required.

Stage-wise Discriminator Function Generation. Given the language instruction of the task, we use visual prompting with OpenAI o3 model [28] to generate the number of semantic manipulation skill stages and corresponding Python functions for each stage-wise discriminator $d_i^{(s)}(s_t)$. Following prior works [106, 107], we leverage VLMs to specify structured relations (e.g., relative distances, alignment conditions) as symbolic arithmetic expressions, rather than manipulating numerical values directly. This approach improves the generalizability and interpretability of the discriminators.

4.2 Synthetic Data Generation for Robust Skill Policies Learning

To improve the robustness and generalization of each skill policy, we leverage simulation to train RL policies under diverse and augmented conditions. For each skill σ_i, we construct a realistic simulation environment, augment its initial and terminal state distributions as well as physical parameters, transfer the corresponding real-world segments into simulation by state estimation, train a base imitation learning policy $\pi_{\sigma_i}^{\text{base}}$ using the transferred segment data, and train a residual RL policy $\pi_{\sigma_i}^{\text{res}}$ complementing the base policy. We then collect simulated demonstrations under real-world observation space and co-train the final policy π_{σ_i} using both real and simulated data.

Real-to-Sim Transfer. For each skill σ_i, we construct a simulation environment that closely matches the real-world scene. To achieve this, we first use existing off-the-shelf 3D reconstruction models [108] to generate textured object meshes from multi-view images. For articulated objects, following prior work [56], we manually separate each mesh into individual links and define their kinematic relationships by adding articulations. To model physical parameters (e.g., mass, friction), we avoid explicit system identification, which typically requires extensive real-world interaction [109, 110]. Instead, we adopt domain randomization: during training, dynamics parameters are sampled from a predefined range, which is effective for sim-to-real transfer. From expert demonstrations τ_{σ_i}, we extract the observed initial and terminal states for each skill σ_i, denoted as $\mathcal{I}_i^{\text{real}}$ and $\mathcal{E}_i^{\text{real}}$, respectively. These correspond to the observed states at the boundaries of skill segments. To improve robustness, we augment both sets via object-centric perturbations (e.g., pose noise, small translations, and rotations. See Appendix for details), yielding: $\mathcal{I}_i^{\text{aug}} = \text{Augment}(\mathcal{I}_i^{\text{real}}), \mathcal{E}_i^{\text{aug}} = \text{Augment}(\mathcal{E}_i^{\text{real}})$. We use FoundationPose [111] to estimate the 6D poses of the objects involved for real-to-sim demonstration transfer.

Skill Policies Training. Given the simulation environment for each skill σ_i and converted demonstration segments from τ_{σ_i}, the goal is to train a robust policy π_{σ_i} within the corresponding initial distribution $\mathcal{I}_i^{\text{aug}}$ and end distribution $\mathcal{E}_i^{\text{aug}}$. Specifically, we first use the converted demonstration segments to train a base policy $\pi_{\sigma_i}^{\text{base}}$ with Behavior Cloning (BC), i.e., $\pi_{\sigma_i}^{\text{base}} = \arg\max_{\pi_{\sigma_i}^{\text{base}}} \mathbb{E}_{(s_t,a_t)} [\log \pi_{\text{base}}(a_t \mid s_t)]$, and then train a residual policy $\pi_{\sigma_i}^{\text{res}}$ using PPO [112] with privileged state information and binary sparse reward. During RL training, the actions from both policies are combined $\pi_{\sigma_i}^{\text{base}}(s_t) + \pi_{\sigma_i}^{\text{res}}(s_t)$. We use orthogonal initialization [113] for the residual policy network and progressive exploration schedule [114] to stabilize training (see Appendix for more details). After that, we rollout the trained policy initialized from $\mathcal{I}_i^{\text{aug}}$ to collect successful trajectories (end within $\mathcal{E}_i^{\text{aug}}$) with real-world observable state information in simulation. Finally, we co-train a policy π_{σ_i} with the simulation trajectories and real-world trajectories using the same BC loss. Notably, real-world data is used to both enable base policy training via state estimation and co-train with simulated data for enhanced robustness.

4.3 Skill Composition via Skill Routing Transformer (SRT) Policy

Given the trained skill policies π_{σ_i}, we aim to chain them together to achieve long-horizon manipulation. To transfer from the terminal state of one skill to the initial state of the succeeding skill, a direct solution is to use motion planning in the real world. However, such method faces challenges in selecting reachable goals between the state distributions $\mathcal{E}_{i-1}^{\text{aug}}$ and $\mathcal{I}_i^{\text{aug}}$ which may require additional human-designed rules, and its computational expense often limits execution speed, especially in cluttered dexterous manipulation scenarios. Instead, we generate diverse and physi-

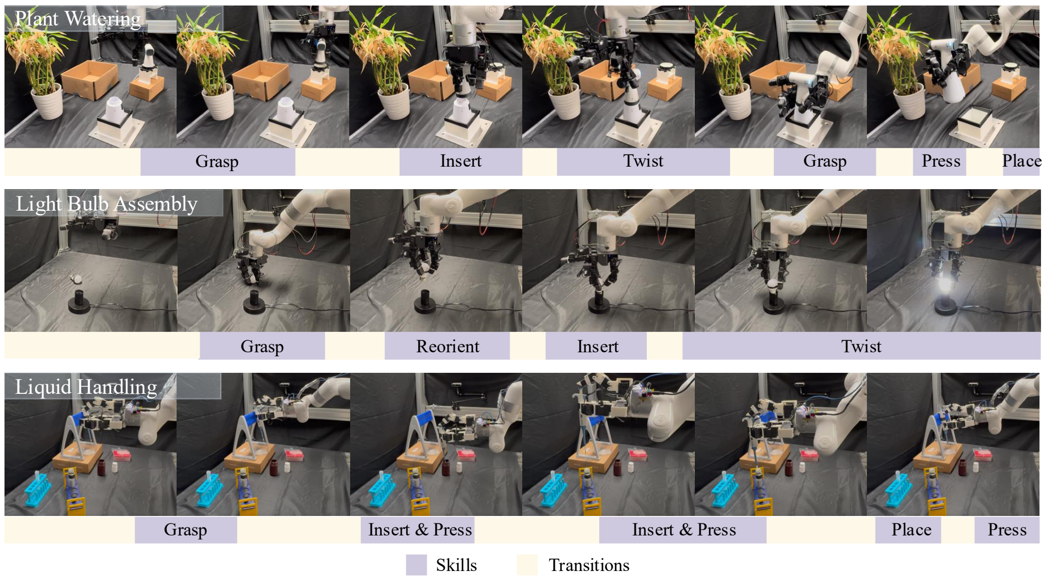

Figure 3: **Real-world Rollout Visualization.**We evaluate LODESTAR on three challenging real-world tasks (Plant Watering, Light Bulb Assembly, and Liquid Handling). The figure visualizes the segmentation results produced by our skill segmentation pipeline on human demonstration trajectories. Skill names are automatically generated by a vision-language model.

cally plausible trajectories in simulation. During data generation, we randomly sample state pairs $(s^{\text{end}}, s^{\text{start}}) \sim \mathcal{E}_{i-1}^{\text{aug}} \times \mathcal{I}_i^{\text{aug}}$, and generate smooth trajectories between them via motion planning. We then filter out the infeasible ones to ensure all trajectories reach targets without collision. This yields a transition dataset $\mathcal{D}^{\text{trans}}$ containing tuples (s_t, a_t, k_t), where $k_t \in \{\texttt{transition}, \texttt{skill}_i\}$ denotes the execution stage.

We then use this dataset to train a transformer-based policy *Skill Routing Transformer (SRT)* policy, which predicts the current transition motion and the stage (select either transition or one skill) for the next timestep. SRT policy $\pi^{\text{g}} : \mathcal{O} \to \mathcal{A} \times \mathcal{K}$ maps the current observation o_t to a dense action $a_t \in \mathcal{A}$ and a discrete stage $k_t \in \mathcal{K}$, where $\mathcal{K} = \{\texttt{transition}\} \cup \{\texttt{skill}_i\}_{i=1}^{K}$. In `transition` stage, the policy outputs low-level actions to bridge between two skills. In $\texttt{skill}_i$ stage, it executes the trained skill policy π_{σ_i}. We implement π^{g} using a Transformer-based architecture over a history of past observations, enabling temporal context awareness in mode prediction and action generation.

5 Experimental Evaluation

We answer the following research questions through experiments: **Q1.** Does LODESTAR lead to better transfer performance and robustness using the generated synthetic data? **Q2.** Does LODESTAR help chaining multiple skill policies to accomplish long-horizon tasks? **Q3.** Can LODESTAR help to improve policy generalization under out-of-distribution (OOD) conditions?

Experimental Setup We evaluate our framework on 3 challenging real-world dexterous manipulation tasks requiring both fine-grained precision and reactivity (as shown in Fig. 3): 1) **Liquid Handling** involves the robot picking up a pipette from a rack, aspirating liquid from a reagent bottle, dispensing it into a test tube, returning the pipette to the rack, and disposing of the used tip. 2) **Plant Watering** requires grasping a spray nozzle, positioning it onto a bottle, securely twisting it in place, and triggering it to water the plants while holding the bottle. 3) **Light Bulb Assembly** tasks the robot with gripping a light bulb, reorienting the bulb in-hand for insertion, and precisely screwing the bulb into a base until lighting. Our real-world setup uses an xArm7 robot paired with either a three-finger hand or a four-finger LEAP hand [115], and utilizes 2 RealSense D435 cameras. Real world data is collected via human teleoperation using a Rokoko Glove to track finger movements and a VIVE Ultimate Tracker to track wrist pose. We use Isaacgym [116] for the simulation envi-

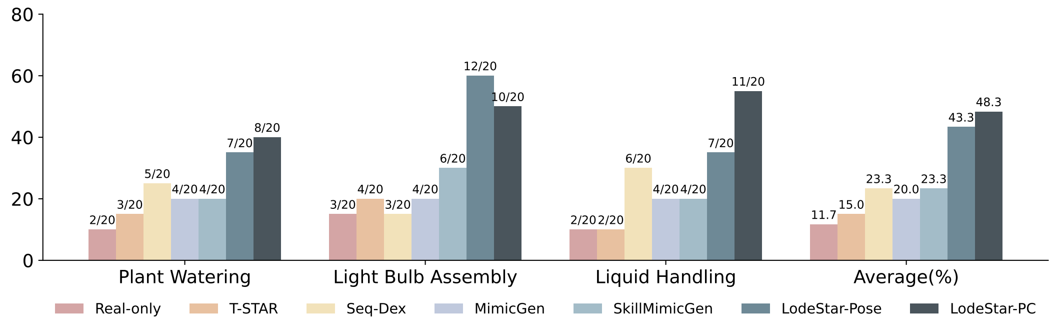

Figure 4: **Success rates across three challenging real-world tasks and their average.** LODESTAR-PC demonstrates superior performance on average and across tasks, boosting average success by at least 25% compared to the baselines.

ronments. See Appendix for the detailed environment setup, system configuration and evaluation protocol.

Baselines and Metrics We compare LODESTAR with the following baselines: 1) **Real-only** uses the same architecture and training as LODESTAR, but is only trained on real-world demonstrations. 2) Skill-chaining baselines: **T-STAR** [23] uses terminal-state regularization, and **Seq-Dex** [25] uses bidirectional optimization. To ensure fairness, we use both simulation and real-world data for these methods. 3) Automatic Data-Generation methods: **MimicGen** [6] employs replay-augmentation on object-centric segments, and **SkillMimicGen** [8] extends this idea to skill-level augmentation with skill chaining via motion planning. We evaluate two variants of our framework, "LODESTAR-POSE" conditions skill policy on estimated initial object poses (as real-time pose estimation is impractical due to its inference overhead), and "LODESTAR-PC" (default option) conditions each skill policy on raw point-cloud observations. Each of the methods evaluates 20 trials. See Appendix for implementation details.

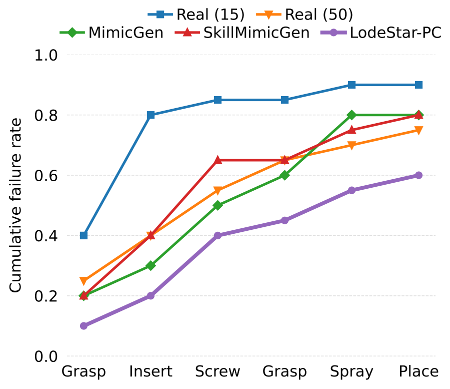

Figure 5: **Cumulative Failure Rate** on the Plant Watering task.

5.1 Results and Analysis

LODESTAR is effective for real-to-sim-to-real transfer performance and robustness (Q1). As illustrated in Fig. 4, LODESTAR-PC achieves the best performance on average and in the three challenging tasks. Compared to the best automatic data-generation baseline SkillMimicGen that uses replay-based demonstration transformation on the object-centric skill segments, LODESTAR-PC boosts the average performance by 25%. We attribute the improvement to two main factors: 1) LODESTAR adopts domain randomization in real-to-sim transfer to learn more robust sim-to-real behavior, while replay-based methods may fail due to the gaps from sensors and the controller in the bidirectional transfer between simulation and reality. 2) LODESTAR uses residual RL on a base policy from imitation learning to correct imperfect demonstrations and generalize across broader state distributions through exploration. On the contrary, replay-based methods may fail to cover a diverse range of initial and terminal skill states because of the kinematic constraints in the source demonstrations. Notably, LODESTAR-POSE, despite its simplicity—conditioning skill policy solely on the estimated initial object pose—performs effectively and even surpasses LODESTAR-PC in the light bulb assembly task, which involves a less cluttered environment than the other two tasks. This suggests that when occlusion is minimal and pose estimation is reliable, LodeStar-Pose serves as a simple yet effective solution.

LodeStar is effective for long-horizon skill chaining (Q2). As shown in Fig. 4, LodeStar significantly outperforms prior skill chaining methods. Compared with T-Star and Seq-Dex, which apply uni- or bi-directional optimization on skill boundaries, LodeStar avoids such regularization for training stabilization and instead employs a more efficient stage transition to chain skills, boosting performance by over 25%. In addition, we observe that online motion planning in SkillMimicGen often fails because of state estimation error from occlusion and sensor perception errors, whereas LodeStar chains skills trained on generated successful synthetic transitions.

LodeStar is effective for improving policy robustness and generalization under OOD situations (Q3). We evaluate policies by initializing with a larger distribution or applying disturbances on the Light Bulb Assembly task. As shown in Tab. 1, although more real-world data (15 demos vs. 50 demos) boosts the performance from 0 to 20% and 5% to 25%, LodeStar achieves ∼2 times higher success rate with only 15 demos and exhibits emergent behavior under disturbances, thereby demonstrating superior robustness and generalization. See Appendix for more details and visualization.

Method (# Demos)	Larger Init Distribution	Disturbances
Real-only (15)	0/20	1/20
Real-only (50)	4/20	5/20
SkillMimicGen (15)	5/20	4/20
LodeStar(15)	**10**/20	**8**/20

Table 1: **Success rates under OOD conditions** on the Light Bulb Assembly task.

Cumulative Failure Rate Analysis. Fig. 5 shows the cumulative failure rates throughout the task execution on the Plant Watering task. LodeStar-PC consistently achieves the lowest failure rate at each stage, demonstrating superior robustness and compounding stability. In contrast, methods trained purely on real-world demonstrations suffer from early failure accumulation, particularly after the "Insert" and "Screw" stage. Both MimicGen and SkillMimicGen reduce early-stage failures but plateau in later stages. These results underscore the effectiveness of our approach in mitigating error accumulation and ensuring reliable execution over extended task horizons.

Ablations. We further evaluate the contribution of key components in LodeStar through ablation studies on the Light Bulb Assembly task, as shown in Tab. 2. Compared to using human predefined skills, our skill segmentation pipeline improves real-world success rate by 20%, highlighting its effectiveness in capturing meaningful skill boundaries. LodeStar also significantly outperforms using RL fine-tuning (by 35% in real world), benefiting from residual RL which stabilizes simulation training. Removing the transition stage or simulation augmentation leads to 20% and 30% drops in real-world success respectively, indicating the importance of smooth skill composition and simulation distribution augmentation. Finally, co-training with real-world data yields an additional 15% improvement over simulation-only training, demonstrating its value in enhancing sim-trained policies. These results again highlight the effectiveness of these key designs in LodeStar.

	Simulation	Real-world
LodeStar	74%	**10/20**
w/ predefined skills	65%	6/20
RL fine-tuning	45%	3/20
w/o transition stage	62%	6/20
w/o sim augmentation	91%	4/20
w/o real co-training	76%	7/20

Table 2: **Success Rates with Ablated Components** on the Light Bulb Assembly task.

6 Conclusion

To summarize, we presents LodeStar, a system for automatic skill decomposition and synthetic data generation from human demonstrations for long-horizon dexterous manipulation tasks. The sim-augmented datasets enable robust skill acquisition, while a global policy integrates and chains these skills into long-horizon executions, enhancing robustness and generalization in real-world environments. The results on three real-world challenging tasks show that LodeStar achieves 2 times higher success rate compared to the best baseline, while also generalizing to a broader range of environment variations. These highlight the potential of combining structured task representations with scalable synthetic data augmentation for efficient and generalizable dexterous robot learning.

7 Limitations.

Despite compelling results, our approach has several limitations, which also highlight multiple avenues for future works: First, while our method demonstrates significant improvements over baseline approaches, the overall task performance has not yet reached saturation and exhibits room for further enhancement. Future research could investigate techniques like human-in-the-loop online correction mechanisms to potentially boost performance levels. Second, the depth sensors used for constructing 3D point clouds, keypoint projection, and pose estimation can be unreliable—particularly when dealing with transparent or highly reflective objects (in our experiments, we manually applied opaque tape to transparent test tubes). Future work might integrate additional sensing modalities, such as tactile contact information, to improve both skill decomposition and policy learning. Third, our real-to-sim pipeline currently overlooks dynamic parameters. Future efforts could apply system-identification techniques to model dynamics in simulation more precisely, thereby enhancing the fidelity of sim-to-real transfer. Lastly, all experiments have been conducted using rigid objects. We did not investigate tasks involving deformable objects, as these objects remain challenging to reconstruct and simulate accurately. Extending our framework with advanced simulation and representation methods for deformable objects would broaden its applicability.

References

[1] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[2] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022.

[3] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.

[4] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard. Recent advances in robot learning from demonstration. *Annual review of control, robotics, and autonomous systems*, 3 (1):297–330, 2020.

[5] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[6] A. Mandlekar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. In *7th Annual Conference on Robot Learning*, 2023.

[7] Z. Jiang, Y. Xie, K. Lin, Z. Xu, W. Wan, A. Mandlekar, L. Fan, and Y. Zhu. Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning. *arXiv preprint arXiv:2410.24185*, 2024.

[8] C. Garrett, A. Mandlekar, B. Wen, and D. Fox. Skillmimicgen: Automated demonstration generation for efficient skill learning and deployment. *arXiv preprint arXiv:2410.18907*, 2024.

[9] Z. Xue, S. Deng, Z. Chen, Y. Wang, Z. Yuan, and H. Xu. Demogen: Synthetic demonstration generation for data-efficient visuomotor policy learning. *arXiv preprint arXiv:2502.16932*, 2025.

[10] Y. Xu, W. Wan, J. Zhang, H. Liu, Z. Shan, H. Shen, R. Wang, H. Geng, Y. Weng, J. Chen, et al. Unidexgrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4737–4746, 2023.

[11] 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.

[12] H.-S. Fang, H. Yan, Z. Tang, H. Fang, C. Wang, and C. Lu. Anydexgrasp: General dexterous grasping for different hands with human-level learning efficiency. *arXiv preprint arXiv:2502.16420*, 2025.

[13] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[14] 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.

[15] 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.

[16] G. Konidaris and A. Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. *Advances in neural information processing systems*, 22, 2009.

[17] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 31(3):360–375, 2012.

[18] A. Clegg, W. Yu, J. Tan, C. K. Liu, and G. Turk. Learning to dress: Synthesizing human dressing motion via deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 37 (6):1–10, 2018.

[19] A. Bagaria and G. Konidaris. Option discovery using deep skill chaining. In *International Conference on Learning Representations*, 2019.

[20] Y. Lee, J. Yang, and J. J. Lim. Learning to coordinate manipulation skills via skill behavior diversification. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ryxB2lBtvH.

[21] X. B. Peng, M. Chang, G. Zhang, P. Abbeel, and S. Levine. Mcp: Learning composable hierarchical control with multiplicative compositional policies. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 3681–3692. Curran Associates, Inc., 2019.

[22] J. Gu, D. S. Chaplot, H. Su, and J. Malik. Multi-skill mobile manipulation for object rearrangement. *arXiv preprint arXiv:2209.02778*, 2022.

[23] Y. Lee, J. J. Lim, A. Anandkumar, and Y. Zhu. Adversarial skill chaining for long-horizon robot manipulation via terminal state regularization. *arXiv preprint arXiv:2111.07999*, 2021.

[24] Z. Chen, Z. Ji, J. Huo, and Y. Gao. Scar: Refining skill chaining for long-horizon robotic manipulation via dual regularization. *Advances in Neural Information Processing Systems*, 37:111679–111714, 2024.

[25] 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.

[26] N. Karaev, I. Rocco, B. Graham, N. Neverova, A. Vedaldi, and C. Rupprecht. Cotracker: It is better to track together. In *European Conference on Computer Vision*, pages 18–35. Springer, 2024.

[27] L. Tang, M. Jia, Q. Wang, C. P. Phoo, and B. Hariharan. Emergent correspondence from image diffusion. *Advances in Neural Information Processing Systems*, 36:1363–1389, 2023.

[28] OpenAI. OpenAI o3: Introducing a new generation of reasoning models. https://openai.com/index/introducing-o3-and-o4-mini/, 2025.

[29] J. K. Salisbury and J. J. Craig. Articulated hands: Force control and kinematic issues. *The International journal of Robotics research*, 1(1):4–17, 1982.

[30] I. Mordatch, Z. Popović, and E. Todorov. Contact-invariant optimization for hand manipulation. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 137–144, 2012.

[31] 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.

[32] Z. He, B. Ai, Y. Liu, W. Wan, H. I. Christensen, and H. Su. Learning dexterous deformable object manipulation through cross-embodiment dynamics learning. In *3rd RSS Workshop on Dexterous Manipulation: Learning and Control with Diverse Data*, 2025.

[33] R. Fearing. Implementing a force strategy for object re-orientation. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 96–102. IEEE, 1986.

[34] L. Han and J. C. Trinkle. Dextrous manipulation by rolling and finger gaiting. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, volume 1, pages 730–735. IEEE, 1998.

[35] D. Rus. In-hand dexterous manipulation of piecewise-smooth 3-d objects. *The International Journal of Robotics Research*, 18(4):355–381, 1999.

[36] Y. Bai and C. K. Liu. Dexterous manipulation using both palm and fingers. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1560–1565. IEEE, 2014.

[37] W. Wan, H. Geng, Y. Liu, Z. Shan, Y. Yang, L. Yi, and H. Wang. Unidexgrasp++: Improving dexterous grasping policy learning via geometry-aware curriculum and iterative generalist-specialist learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3891–3902, 2023.

[38] 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.

[39] Y. Liu, Y. Liu, C. Jiang, K. Lyu, W. Wan, H. Shen, B. Liang, Z. Fu, H. Wang, and L. Yi. Hoi4d: A 4d egocentric dataset for category-level human-object interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21013–21022, 2022.

[40] K. Shaw, Y. Li, J. Yang, M. K. Srirama, R. Liu, H. Xiong, R. Mendonca, and D. Pathak. Bimanual dexterity for complex tasks. *arXiv preprint arXiv:2411.13677*, 2024.

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

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

[43] 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.

[44] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 4243–4250. IEEE, 2018.

[45] S. Nasiriany, A. Maddukuri, L. Zhang, A. Parikh, A. Lo, A. Joshi, A. Mandlekar, and Y. Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024.

[46] J. Wang, Y. Qin, K. Kuang, Y. Korkmaz, A. Gurumoorthy, H. Su, and X. Wang. Cyberdemo: Augmenting simulated human demonstration for real-world dexterous manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17952–17963, 2024.

[47] L. Wang, J. Zhao, Y. Du, E. H. Adelson, and R. Tedrake. Poco: Policy composition from and for heterogeneous robot learning. *arXiv preprint arXiv:2402.02511*, 2024.

[48] L. Ankile, A. Simeonov, I. Shenfeld, M. Torne, and P. Agrawal. From imitation to refinement–residual rl for precise assembly. *arXiv preprint arXiv:2407.16677*, 2024.

[49] A. Maddukuri, Z. Jiang, L. Y. Chen, S. Nasiriany, Y. Xie, Y. Fang, W. Huang, Z. Wang, Z. Xu, N. Chernyadev, et al. Sim-and-real co-training: A simple recipe for vision-based robotic manipulation. *arXiv preprint arXiv:2503.24361*, 2025.

[50] A. Wei, A. Agarwal, B. Chen, R. Bosworth, N. Pfaff, and R. Tedrake. Empirical analysis of sim-and-real cotraining of diffusion policies for planar pushing from pixels. *arXiv preprint arXiv:2503.22634*, 2025.

[51] H. Geng, F. Wang, S. Wei, Y. Li, B. Wang, B. An, C. T. Cheng, H. Lou, P. Li, Y.-J. Wang, et al. Roboverse: Towards a unified platform, dataset and benchmark for scalable and generalizable robot learning. *arXiv preprint arXiv:2504.18904*, 2025.

[52] H. Liu, W. Wan, X. Yu, M. Li, J. Zhang, B. Zhao, Z. Chen, Z. Wang, Z. Zhang, and H. Wang. Navid-4d: Unleashing spatial intelligence in egocentric rgb-d videos for vision-and-language navigation. 2025.

[53] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The international journal of Robotics Research*, 31(5):647–663, 2012.

[54] M. Tancik, E. Weber, E. Ng, R. Li, B. Yi, T. Wang, A. Kristoffersen, J. Austin, K. Salahi, A. Ahuja, et al. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 conference proceedings*, pages 1–12, 2023.

[55] Z. Jiang, C.-C. Hsu, and Y. Zhu. Ditto: Building digital twins of articulated objects from interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5616–5626, 2022.

[56] 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 preprint arXiv:2403.03949*, 2024.

[57] S. Patel, X. Yin, W. Huang, S. Garg, H. Nayyeri, L. Fei-Fei, S. Lazebnik, and Y. Li. A real-to-sim-to-real approach to robotic manipulation with vlm-generated iterative keypoint rewards. *arXiv preprint arXiv:2502.08643*, 2025.

[58] W. Ye, F. Liu, Z. Ding, Y. Gao, O. Rybkin, and P. Abbeel. Video2policy: Scaling up manipulation tasks in simulation through internet videos. *arXiv preprint arXiv:2502.09886*, 2025.

[59] H. Xia, E. Su, M. Memmel, A. Jain, R. Yu, N. Mbiziwo-Tiapo, A. Farhadi, A. Gupta, S. Wang, and W.-C. Ma. Drawer: Digital reconstruction and articulation with environment realism. *arXiv preprint arXiv:2504.15278*, 2025.

[60] Z. Chen, A. Walsman, M. Memmel, K. Mo, A. Fang, K. Vemuri, A. Wu, D. Fox, and A. Gupta. Urdformer: A pipeline for constructing articulated simulation environments from real-world images. *arXiv preprint arXiv:2405.11656*, 2024.

[61] Y. Wang, Z. Xian, F. Chen, T.-H. Wang, Y. Wang, K. Fragkiadaki, Z. Erickson, D. Held, and C. Gan. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation. *arXiv preprint arXiv:2311.01455*, 2023.

[62] T. Dai, J. Wong, Y. Jiang, C. Wang, C. Gokmen, R. Zhang, J. Wu, and L. Fei-Fei. Automated creation of digital cousins for robust policy learning. *arXiv preprint arXiv:2410.07408*, 2024.

[63] Z. Mandi, Y. Weng, D. Bauer, and S. Song. Real2code: Reconstruct articulated objects via code generation. *arXiv preprint arXiv:2406.08474*, 2024.

[64] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[65] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[66] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019.

[67] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull. Active domain randomization. In *Conference on Robot Learning*, pages 1162–1176. PMLR, 2020.

[68] A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

[69] B. Evans, A. Thankaraj, and L. Pinto. Context is everything: Implicit identification for dynamics adaptation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2642–2648. IEEE, 2022.

[70] F. Muratore, T. Gruner, F. Wiese, B. Belousov, M. Gienger, and J. Peters. Neural posterior domain randomization. In *Conference on robot learning*, pages 1532–1542. PMLR, 2022.

[71] P. Huang, X. Zhang, Z. Cao, S. Liu, M. Xu, W. Ding, J. Francis, B. Chen, and D. Zhao. What went wrong? closing the sim-to-real gap via differentiable causal discovery. In *Conference on Robot Learning*, pages 734–760. PMLR, 2023.

[72] A. Z. Ren, H. Dai, B. Burchfiel, and A. Majumdar. Adaptsim: Task-driven simulation adaptation for sim-to-real transfer. *arXiv preprint arXiv:2302.04903*, 2023.

[73] 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.

[74] K. Rao, C. Harris, A. Irpan, S. Levine, J. Ibarz, and M. Khansari. Rl-cyclegan: Reinforcement learning aware simulation-to-real. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11157–11166, 2020.

[75] D. Ho, K. Rao, Z. Xu, E. Jang, M. Khansari, and Y. Bai. Retinagan: An object-aware approach to sim-to-real transfer. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10920–10926. IEEE, 2021.

[76] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[77] G. Konidaris. On the necessity of abstraction. *Current opinion in behavioral sciences*, 29: 1–7, 2019.

[78] Y. Lee, S.-H. Sun, S. Somasundaram, E. S. Hu, and J. J. Lim. Composing complex skills by learning transition policies. In *International conference on learning representations*, 2019.

[79] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

[80] J. Oh, S. Singh, H. Lee, and P. Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. In *International Conference on Machine Learning*, pages 2661–2670. PMLR, 2017.

[81] Y. Zhu, J. Tremblay, S. Birchfield, and Y. Zhu. Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6541–6548. Ieee, 2021.

[82] T. Silver, A. Athalye, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling. Learning neuro-symbolic skills for bilevel planning. *arXiv preprint arXiv:2206.10680*, 2022.

[83] S. Nasiriany, H. Liu, and Y. Zhu. Augmenting reinforcement learning with behavior primitives for diverse manipulation tasks. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7477–7484. IEEE, 2022.

[84] S. Cheng and D. Xu. League: Guided skill learning and abstraction for long-horizon manipulation. *IEEE Robotics and Automation Letters*, 8(10):6451–6458, 2023.

[85] Y. Guo, B. Tang, I. Akinola, D. Fox, A. Gupta, and Y. Narang. Srsa: Skill retrieval and adaptation for robotic assembly tasks. *arXiv preprint arXiv:2503.04538*, 2025.

[86] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *2009 IEEE international conference on robotics and automation*, pages 763–768. IEEE, 2009.

[87] Z. Su, O. Kroemer, G. E. Loeb, G. S. Sukhatme, and S. Schaal. Learning manipulation graphs from demonstrations using multimodal sensory signals. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 2758–2765. IEEE, 2018.

[88] D. Xu, S. Nair, Y. Zhu, J. Gao, A. Garg, L. Fei-Fei, and S. Savarese. Neural task programming: Learning to generalize across hierarchical tasks. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3795–3802. IEEE, 2018.

[89] T. Kipf, Y. Li, H. Dai, V. Zambaldi, A. Sanchez-Gonzalez, E. Grefenstette, P. Kohli, and P. Battaglia. Compile: Compositional imitation learning and execution. In *International Conference on Machine Learning*, pages 3418–3428. PMLR, 2019.

[90] Y. Zhu, P. Stone, and Y. Zhu. Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation. *IEEE Robotics and Automation Letters*, 7(2):4126–4133, 2022.

[91] S. Li, Z. Huang, T. Chen, T. Du, H. Su, J. B. Tenenbaum, and C. Gan. Dexdeform: Dexterous deformable object manipulation with human demonstrations and differentiable physics. *arXiv preprint arXiv:2304.03223*, 2023.

[92] M. Xu, Z. Xu, C. Chi, M. Veloso, and S. Song. Xskill: Cross embodiment skill discovery. In *Conference on robot learning*, pages 3536–3555. PMLR, 2023.

[93] W. Wan, Y. Zhu, R. Shah, and Y. Zhu. Lotus: Continual imitation learning for robot manipulation through unsupervised skill discovery. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 537–544. IEEE, 2024.

[94] Z. Wang, J. Hu, C. Chuck, S. Chen, R. Martín-Martín, A. Zhang, S. Niekum, and P. Stone. Skild: Unsupervised skill discovery guided by factor interactions. *arXiv preprint arXiv:2410.18416*, 2024.

[95] J. Schmidhuber. *Towards compositional learning with dynamic neural networks*. Inst. für Informatik, 1990.

[96] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International conference on machine learning*, pages 3540–3549. PMLR, 2017.

[97] P.-L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[98] O. Nachum, S. S. Gu, H. Lee, and S. Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

[99] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

[100] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

[101] U. A. Mishra, S. Xue, Y. Chen, and D. Xu. Generative skill chaining: Long-horizon skill planning with diffusion models. In *Conference on Robot Learning*, pages 2905–2925. PMLR, 2023.

[102] U. A. Mishra, Y. Chen, and D. Xu. Generative factor chaining: Coordinated manipulation with diffusion-based factor graph. In *8th Annual Conference on Robot Learning*, 2024.

[103] S. Haldar and L. Pinto. Point policy: Unifying observations and actions with key points for robot manipulation. *arXiv preprint arXiv:2502.20391*, 2025.

[104] P. Sundaresan, H. Hu, Q. Vuong, J. Bohg, and D. Sadigh. What's the move? hybrid imitation learning via salient points. *arXiv preprint arXiv:2412.05426*, 2024.

[105] Z. Zhang, Y. Li, O. Bastani, A. Gupta, D. Jayaraman, Y. J. Ma, and L. Weihs. Universal visual decomposer: Long-horizon manipulation made easy. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6973–6980. IEEE, 2024.

[106] W. Huang, C. Wang, Y. Li, R. Zhang, and L. Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *arXiv preprint arXiv:2409.01652*, 2024.

[107] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

[108] AR Code. AR Code. https://ar-code.com/, 2022.

[109] J. Bohg, K. Hausman, B. Sankaran, O. Brock, D. Kragic, S. Schaal, and G. S. Sukhatme. Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 33(6):1273–1291, 2017.

[110] N. Pfaff, E. Fu, J. Binagia, P. Isola, and R. Tedrake. Scalable real2sim: Physics-aware asset generation via robotic pick-and-place setups. *arXiv preprint arXiv:2503.00370*, 2025.

[111] B. Wen, W. Yang, J. Kautz, and S. Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17868–17879, 2024.

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

[113] A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[114] X. Yuan, T. Mu, S. Tao, Y. Fang, M. Zhang, and H. Su. Policy decorator: Model-agnostic online refinement for large policy model. *arXiv preprint arXiv:2412.13630*, 2024.

[115] K. Shaw, A. Agarwal, and D. Pathak. Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning. *Robotics: Science and Systems (RSS)*, 2023.

[116] 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.

[117] R. Ding, Y. Qin, J. Zhu, C. Jia, S. Yang, R. Yang, X. Qi, and X. Wang. Bunny-visionpro: Real-time bimanual dexterous teleoperation for imitation learning. 2024. URL `https://arxiv.org/abs/2407.03162`.

[118] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3651–3657. IEEE, 2019.

[119] M. Ahn, H. Zhu, K. Hartikainen, H. Ponte, A. Gupta, S. Levine, and V. Kumar. Robel: Robotics benchmarks for learning with low-cost robots. In *Conference on robot learning*, pages 1300–1313. PMLR, 2020.

[120] 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.

[121] C. Wang, H. Shi, W. Wang, R. Zhang, L. Fei-Fei, and C. K. Liu. Dexcap: Scalable and portable mocap data collection system for dexterous manipulation. *arXiv preprint arXiv:2403.07788*, 2024.

[122] Y. Qin, W. Yang, B. Huang, K. Van Wyk, H. Su, X. Wang, Y.-W. Chao, and D. Fox. Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system. In *Robotics: Science and Systems*, 2023.

[123] K. Zakka. Mink: Python inverse kinematics based on MuJoCo, July 2024. URL `https://github.com/kevinzakka/mink`.

[124] GitHub - xArm-Developer/xarm_ros2: ROS2 developer packages for robotic products from UFACTORY — github.com. `https://github.com/xArm-Developer/xarm_ros2`. [Accessed 04-05-2025].

[125] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. URL `https://arxiv.org/abs/2408.00714`.

[126] C. Wang, H. Fang, H.-S. Fang, and C. Lu. Rise: 3d perception makes real-world robot imitation simple and effective. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2870–2877. IEEE, 2024.

[127] 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.

[128] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[129] B. Sundaralingam, S. K. S. Hari, A. Fishman, C. Garrett, K. Van Wyk, V. Blukis, A. Millane, H. Oleynikova, A. Handa, F. Ramos, et al. curobo: Parallelized collision-free minimum-jerk robot motion generation. *arXiv preprint arXiv:2310.17274*, 2023.

[130] S. Haldar, Z. Peng, and L. Pinto. Baku: An efficient transformer for multi-task policy learning. *arXiv preprint arXiv:2406.07539*, 2024.

[131] A. Karpathy. minGPT: A minimal PyTorch reimplementation of the OpenAI GPT. `https://github.com/karpathy/minGPT`, 2021.

Supplementary Material

A Real Robot Platform

In this section, we provide details about our real robot platform, including the hardware setup, teleoperation system, and environment setup.

A.1 Hardware Setup

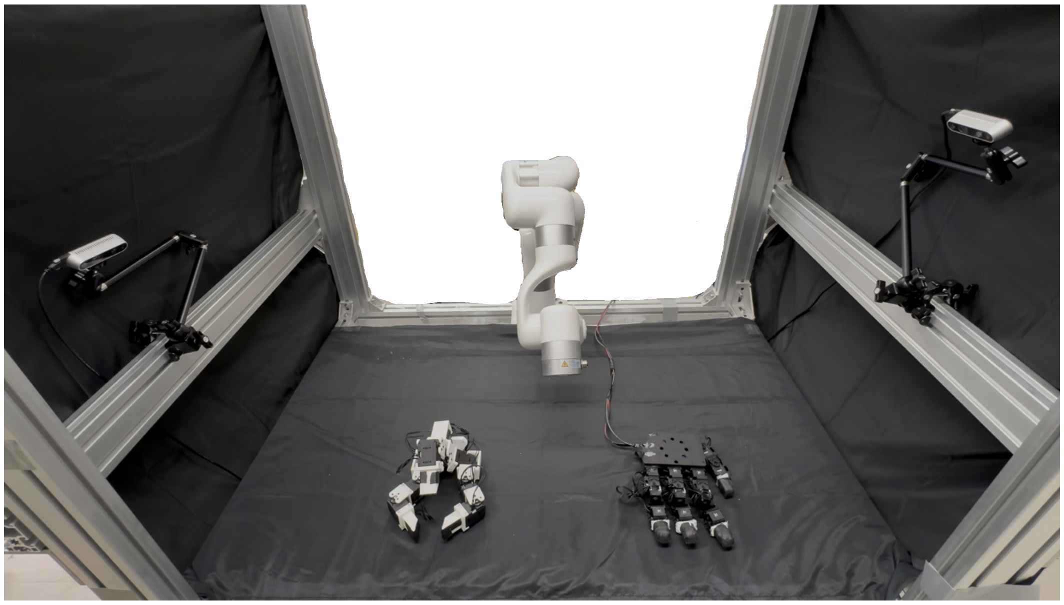

Figure 6: **Hardware Setup for LODESTAR**. The hardware setup uses an xArm7 robot arm with two RealSense D435 cameras. We use two different dexterous robotic hands: **(Left)** a three-fingered hand with 9 degrees of freedom (DoF), and **Right** a four-fingered Leap hand with 16 DoF.

As shown in Fig. 6, our hardware system consists of an xArm7 robot mounted on the tabletop. We use either a three-finger hand or a four-finger LEAP hand [115] at the end-effector of the xArm7 robot. There are two RealSense D435 cameras mounted on the left and right sides of the workspace, respectively.

A.1.1 Robot Arm

We use a joint velocity controller from BunnyVisionPro [117] to control the xArm7 robot. The controller receives joint position commands at 20 Hz. These commanded positions are used by a Proportional-Derivative (PD) controller to compute corresponding joint velocities. For safety, these joint velocities are subsequently clipped before being used to command the robot's motors at a rate of 250 Hz.

A.1.2 Dexterous Hand

For the Liquid Handling task, we use a three-finger hand as shown in Fig. 7. For the Plant Watering and the Light Bulb Assembly tasks, we use a four-finger LEAP hand as shown in Fig. 8.

The three-finger hand is inspired by the DClaw robot [118, 119]. It consists of 9 Dynamixel XL430-W250-T servo motors and has 9 degrees of freedom (DOF) correspondingly. Compared to the DClaw robot, we replace the metal connectors with 3D printed parts for a lower price and change

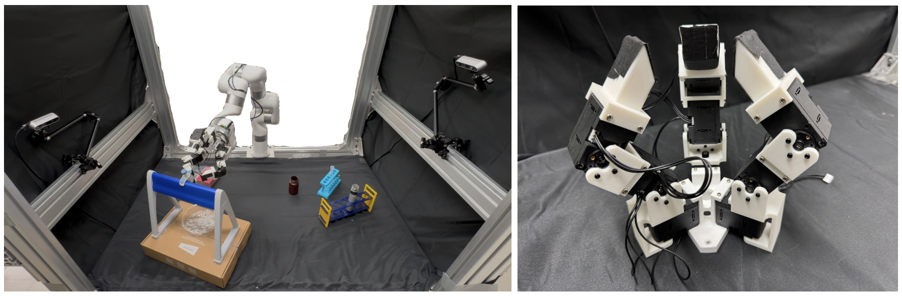

Figure 7: **Three-finger Hand Used for the Liquid Handling Task**. **Left**: The three-finger hand is approaching the pipette. **Right**: The three-finger hand consists of 9 degrees of freedom and uses a flat surface at the fingertips for better gripping performance.

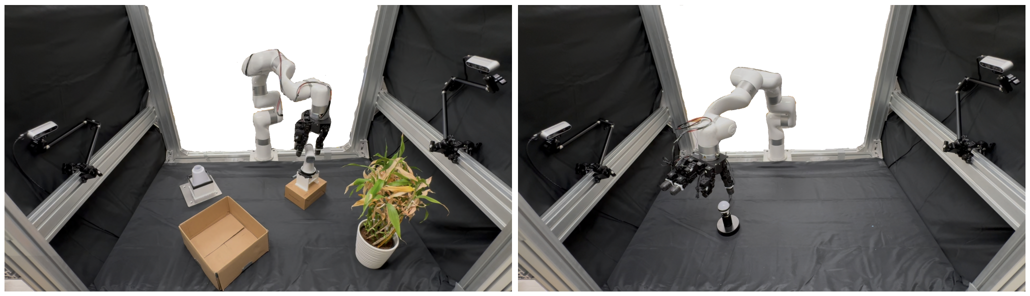

Figure 8: **LEAP Hand Used for the Plant Watering and Light Bulb Assembly Tasks**. **Left**: The LEAP hand is using the thumb, index, and middle fingers to grasp the spray nozzle. **Right**: The LEAP hand is using the thumb and index fingers to screw the bulb in place.

the finger tips from a sphere to a flat surface for better gripping performance. The design of the three-finger hand will be open-sourced on the paper website.

We use a light version of the LEAP hand that comprises 16 Dynamixel XL330-M288-T motors and has 16 DOF correspondingly.

We use a joint position controller for both the three-finger hand and the LEAP hand. The controller receives joint position commands at 20 Hz and sends the commands to the motors at 60 Hz. Since Dynamixel motors have a PD controller internally, we opt not to do interpolation externally. Furthermore, friction tape is applied to the fingertips of both hands to enhance friction and improve object grasping.

A.1.3 Obtaining Point Cloud from Multi-view Cameras

For all three manipulation tasks, we use two RealSense D435 cameras for point cloud reconstruction to address occlusion issues. In detail, we determine the extrinsics of the cameras in the robot base coordinate by calibrating them with a ChArUco board. Then, the captured colorized point clouds are transformed into the robot base coordinate and concatenated. We crop the concatenated point clouds within the robot's workspace. Additionally, to address the sim-to-real gap from sensor noise encountered in the real world, we use Flying Point Augmentation [120], where large Gaussian noise is randomly added to the point clouds with 0.5% probability. One example is shown in Fig. 9.

A.2 Teleoperation System

To collect demonstrations for dexterous manipulation requiring high precision and reactivity, the teleoperation system needs to accurately track finger motions and hand poses. To this end, inspired by DexCap[121], we use a Rokoko Smart glove to track finger joint positions and mount a Vive Ultimate tracker on it to estimate hand poses as shown in Fig. 10. Since the Rokoko glove and the

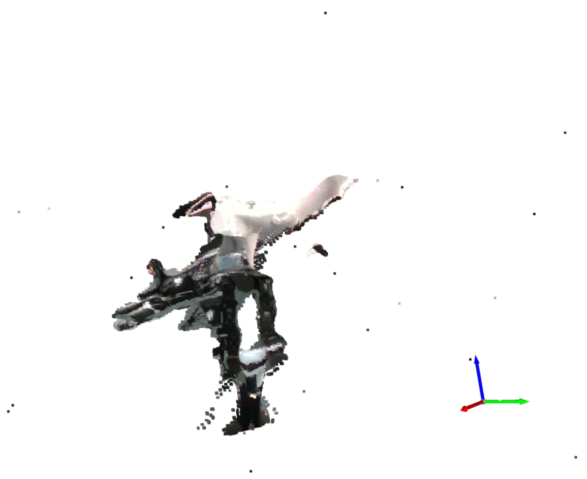

Figure 9: **Visualization of input real-world point clouds**. The point clouds from the two cameras are concatenated in the robot base coordinate and cropped to be within the workspace. We additionally add a Gaussian noise with a small probability to the point clouds to enhance robustness against the sim-to-real gap from sensor noise.

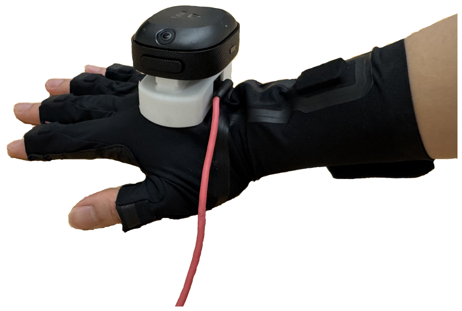

Figure 10: **Teleoperation System for LODESTAR**. We use a Rokoko Smart glove to track finger motions and mount a Vive Ultimate tracker on the glove to track hand poses.

Vive tracker can only stream data to a Windows machine, we forward the tracking data at 50 Hz from the Windows machine to a Ubuntu laptop for controlling the robot.

We apply a low-pass filter to the tracked hand poses and finger joint positions to generate smooth motion for the robot. We directly use the filtered hand pose from the Vive tracker as the target pose of the end effector of the xArm7 robot. We calculate the fingertip positions from human data, linearly transform the positions to account for the larger finger sizes of the robotic hands, and use the transformed positions as the target for the fingertips of the robotic hands [122]. We use mink [123], a differential inverse kinematics library, to calculate the joint positions for both the robotic hands and the xArm7 robot at the same time.

A.3 Environment Setup

In this section, we provide the details of the environment setup for the three dexterous manipulation tasks: Liquid Handling, Plant Watering, and Light Bulb Assembly.

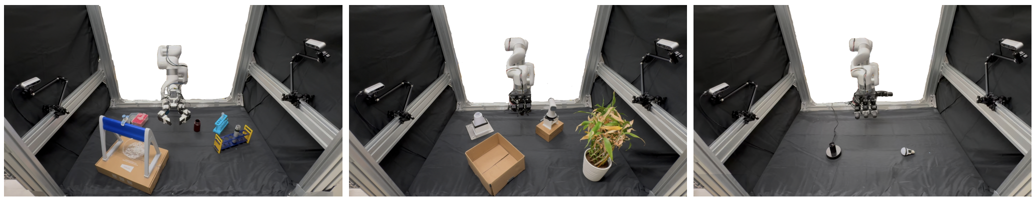

Figure 11: **Environment Setup for the three dexterous manipulation tasks**. **Left**: Liquid Handling. **Middle**: Plant Watering. **Right**: Light Bulb Assembly.

A.3.1 Liquid Handling

In the Liquid Handling task, the tabletop workspace contains: 1) a pipette, 2) a rack, 3) a reagent bottle, 4) a test tube, 5) a container for dispensed tips, and 6) other objects that commonly appear in a biochemical experiment. In this task, the robot needs to

1. grasp the pipette from the rack,
2. aspirate liquid from the reagent bottle,
3. dispense the liquid into the test tube,
4. put the pipette back onto the rack, and
5. dispose of the used tip into the container below.

A.3.2 Plant Watering

In the Plant Watering task, there are 1) a spray nozzle, 2) a spray bottle body, 3) a nozzle stand to hold the spray nozzle vertically, 4) a container to hold the bottle body, 5) an open cardboard box, and 6) a potted plant to be watered. To water the plant, the robot needs to

1. grasp the spray nozzle from the nozzle stand,
2. insert it onto the spray bottle body,
3. securely twist it in place,
4. grasp the assembled spray bottle,
5. press the trigger in front of the plant while holding the bottle, and
6. place the spray bottle into the cardboard box.

A.3.3 Light Bulb Assembly

In the Light Bulb Assembly task, the workspace consists of 1) a light bulb and 2) a bulb base. To accomplish the task, the robot needs to

1. grasp the light bulb,
2. reorient the bulb in-hand for insertion,
3. insert the light bulb into the bulb base,
4. precisely screw the bulb into the base until illumination.

B Simulation Training Details

In this section, we introduce more details about training in simulation, including the used simulator, task designs, skill segmentation, real-to-sim transfer, residual reinforcement learning (residual RL), skill policy training, and skill routing transformer policy.

B.1 The Simulator

We use Isaac Gym [116] as the simulator backend. We use the xArm7 model from its ROS package [124] and the LEAP hand model from [115]. For the three-finger hand, we manually designed its 3D model and exported it to URDF. The 3d models of the objects in the workspace are generated from multi-view images with AR-Code [108]. Inspired by [56], we manually separate the meshes into individual links and define the kinematic relationships for the articulated objects to be manipulated. See more details in Sec. B.3.1.

B.2 Skill Segmentation

We perform frame-level segmentation of long-horizon dexterous demonstrations into multiple skill segments. This section details the keypoint proposal method and the generation of the discriminator function used for segmenting the demonstration trajectories.

B.2.1 Keypoint Proposal

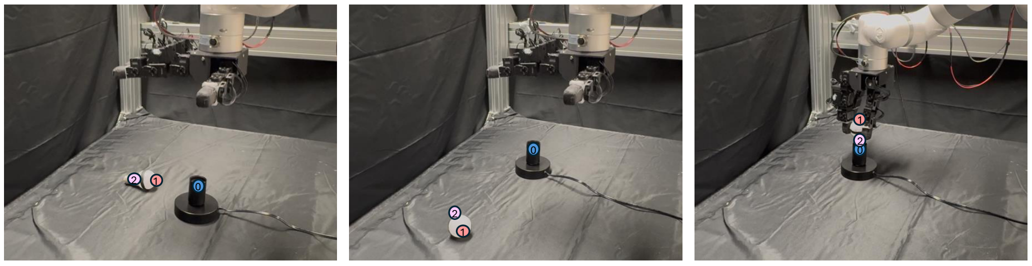

Figure 12: **Process of keypoint semantic correspondence and tracking.** A semantic correspondence model propagates annotated keypoints from the initial frame of one demonstration (**left**) to the initial frame of another (**middle**). Subsequently, a point tracking model tracks these keypoints in further frames of this demonstration (**right**).

For each of the three demonstration tasks (Liquid Handling, Plant Watering, and Light Bulb Assembly), we randomly select a reference demonstration trajectory. With this reference trajectory, we manually annotate N semantically meaningful keypoints $\{\hat{p}_l\}_{l=1}^{N}$ on task-related objects in the first frame. Then, we propagate these initial keypoints to the first frame of other demonstration trajectories with the semantic correspondence model DIFT [27], obtaining consistent keypoints among all the demonstrations. After that, we track the keypoints for each frame from start to end within each trajectory with Co-Tracker [26]. A sample for the process above for each task is visualized in Fig. 12.

B.2.2 Discriminator Function Generation

We design the language instruction for each task and feed it with the first image and the corresponding tracked keypoints into the OpenAI o3 model, which outputs the total number of skill stages and the corresponding discriminator functions $d_i^{(s)}(s_t)$ in Python.

B.3 Real-to-Sim Transfer

Given the segmented demonstration for each skill, we need to create 3D models of the objects and place them at their corresponding positions in the simulation environment. In this section, we show the details for creating the 3d assets and estimating the object poses.

B.3.1 3D Asset Generation

We use AR-Code [108] to create the textured mesh for each object by capturing multi-view images. This will take around 3-5 minutes for each object. We show a full list of the meshes created in

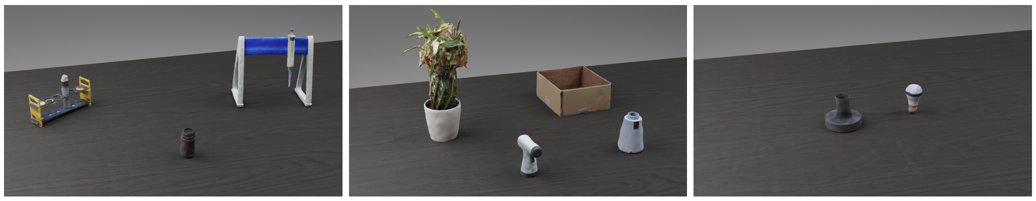
Figure 13: **Textured Meshes Created for the three manipulation tasks**. **Left**: Liquid Handling. **Middle**: Plant Watering. **Right**: Light Bulb Assembly.

Fig. 13. Following the methodology in [56] for articulated objects, we manually decompose their meshes into individual links and establish their kinematic relationships by defining articulations which take around 3-5 minutes for each articulated objects.

B.3.2 State Estimation

Accurate estimation of object poses within the workspace is crucial for seamlessly transferring objects to simulation and generating synthetic augmented trajectories for skill training. We leverage tracked keypoints (see Sec. B.2.1) to prompt SAM2 [125] and obtain a segmentation mask for each object of interest. Using this segmentation mask and the reconstructed 3D mesh, we then employ FoundationPose [111] to estimate the objects' 6D poses.

B.3.3 Domain Randomization

To enhance the robustness against the sim-to-real gap for each skill policy, we apply domain randomization to various simulation properties for the objects involved. The randomization parameters are given in Tab. 3.

Parameter	Type	Distribution	Parameter	Type	Distribution
object mass	Scaling	$\mathcal{U}(0.5,\ 1.5)$	joint lower range	Additive	$\mathcal{N}(0,\ 0.01)$
object static friction	Scaling	$\mathcal{U}(0.7,\ 1.3)$	joint upper range	Additive	$\mathcal{N}(0,\ 0.01)$
robot static friction	Scaling	$\mathcal{U}(0.7,\ 1.3)$	joint damping	Scaling	$\mathcal{E}(0.3,\ 3.0)$
state observation	Additive	$\mathcal{U}(-0.002,\ 0.002)$	joint stiffness	Scaling	$\mathcal{E}(0.75,\ 1.5)$
action	Additive	$\mathcal{N}(0,\ 0.01)$	gravity	Scaling	$\mathcal{U}(0.9,\ 1.1)$
restitution	Scaling	$\mathcal{U}(0.5,\ 1.5)$	Compliance	Scaling	$\mathcal{U}(0.5,\ 1.5)$

Table 3: **Randomization ranges for physical and control parameters.** $\mathcal{U}(a, b)$ denotes uniform distribution over $[a, b]$. $\mathcal{N}(\mu, \sigma^2)$ denotes Gaussian distribution with mean μ and variance σ^2. $\mathcal{E}(a, b)$ denotes exponential-uniform distribution, i.e., $\mathcal{E}(a, b) = \exp(\mathcal{U}(\log a, \log b))$.

B.4 Residual Reinforcement Learning

Given the converted skill segments in simulation, we generate synthetic augmentation data to improve the robustness and generalization for each skill policy. We utilize residual reinforcement learning (residual RL) to create diverse simulated demonstrations. In this section, we introduce the details for the base policy and the residual policy in residual RL.

B.4.1 Base Policy Training

With the skill segments converted into simulation, we first train a state-based base policy using Behavior Cloning. This base policy utilizes a similar diffusion-based architecture in [126] (the vision encoder is replaced with an MLP to encoder object pose) and outputs an action chunk [127, 128] to predict a set of future actions. The observation space of the base policy is shown in Tab. 4 .The detailed parameters for training the base policy are given in Tab. 5.

Name	Dimension	Name	Dimension
Arm joint position	7	Hand joint position	9 / 16
Arm joint velocity	7	Hand joint velocity	9 / 16
Object Pose	N_{obj} * 6	Fingertip pose	N_{finger} * 6

Table 4: **The observation space of base policy.**

Parameter	Value	Parameter	Value
Downsampling dimensions	[256, 512, 1024]	Embedding dimensions	64
Kernel size	5	Observation horizon	2
Prediction horizon	16	Action horizon	8
DDPM training steps	100	DDIM inference steps	4
Gradient steps	200000	Batch size	1024
Learning rate	1e-4	Optimizer	AdamW

Table 5: **State-based diffusion policy hyperparameters.**

B.4.2 Residual Policy Training

We utilize Proximal Policy Optimization (PPO), a model-free RL algorithm, to learn a residual policy complementing the base policy. The residual policy use a simple MLP architecture. The observation space of residual policy is shown in Tab 6. The hyperparameters used in PPO training are given in Tab. 7. In addition, we use orthogonal initialization [113] and progressive exploration schedule [114] for the residual policy learning. The progressive exploration schedule employs an ϵ-greedy strategy where, at each step, the agent uses the residual policy with probability ϵ and otherwise relies solely on the base policy. The probability ϵ increases linearly from 0 to 1 over H environment steps, where $H = 100$K.

Name	Dimension	Name	Dimension
Arm joint position	7	Hand joint position	9 / 16
Arm joint velocity	7	Hand joint velocity	9 / 16
Object states	N_{obj} * 13	fingertip state	N_{finger} * 13
Contact Forces	N_{obj} * 3	-	-

Table 6: **The observation space of residual policy.** The object and fingertip state includes position, orientation, linear velocities and angular velocities.

B.5 Skill Policy Training

We use 1000 generated simulation demonstrations and 15 real world demonstrations to co-train the final skill policy. The observation space of the skill policy is shown in Tab. 8. We adopt a diffusion-based policy architecture similar to that of the base policy. Specifically, for LODESTAR-PC, we utilize the policy architecture from [126], which comprises a 3D encoder built with sparse convolutions for point cloud encoding and a diffusion-based action head. For LODESTAR-POSE, this 3D sparse encoder is replaced with an MLP.

B.6 Skill Routing Transformer (SRT) Policy

Given the trained skill policies, we train a Skill Routing Transformer (SRT) policy to chain them to accomplish the long-horizon task. In this section, we give the details for generating the transition dataset and policy training.

Hyperparameter	**Value**	**Hyperparameter**	**Value**
PPO rollout steps	8	Batches per agent	4
Learning epochs	5	Desired KL	0.16
Episode length	200	Policy iteration threshold	0.005
Discount factor	0.96	GAE parameter	0.95
Entropy coeff.	0.0	PPO clip range	0.2
Learning rate	0.0003	Value loss coeff.	1.0
Max gradient norm	1.0	Initial noise std.	0.8
Clip actions	1.0	Clip observations	5.0

Table 7: **Hyperparameters for RL policy.**

Name	**Dimension**	**Name**	**Dimension**
Arm joint position	7	Hand joint position	9 / 16
Arm joint velocity	7	Hand joint velocity	9 / 16
Initial object pose	N_{obj} * 6	Point num	1024

Table 8: **The observation space of skill policy.** The initial object pose is used for LODESTAR-POSE and the point cloud is used for LODESTAR-PC.

B.6.1 Transition Data Generation

For generating transition trajectories between skill segments, we randomly sample state pairs $(s^{\text{end}}, s^{\text{start}}) \sim \mathcal{E}_{i-1}^{\text{aug}} \times \mathcal{I}_{i}^{\text{aug}}$, where $\mathcal{E}_{i-1}^{\text{aug}}$ is the termination set for the previous skill segment and $\mathcal{I}_{i}^{\text{aug}}$ is the initial set for the subsequent skill segment. For each sampled state pair, we then utilize cuRobo [129] to generate a collision-free transition trajectory. We continue this process until 1000 trajectories have been successfully generated for each specific transition motion between consecutive skills.

B.6.2 Policy Training

For Skill Routing Transformer (SRT) policy, we use similar architecture with BAKU [130], an efficient transformer for multi-task policy learning. The SRT policy architecture is shown in Fig. 14 . We use a 3D encoder built with sparse convolution [126] to encode point cloud input and an MLP to encode the robot state. The observation trunk use minGPT [131] architecture and the action head use an MLP. The action head output an action chunk and stage (to decide either a skill or transition). The parameters of SRT policy is shown in Tab. 9. The SRT policy utilizes the same point cloud observation space as the skill policy that takes point cloud as input, with this observation space detailed in Tab. 8.

Hyperparameter	**Value**	**Hyperparameter**	**Value**
Observation trunk	Transformer	Transformer architecture	minGPT [131]
Transformer hidden dim	256	Observation history length	10
Voxel size (mm)	5	Point feature dim	512
Encoding block	4	Decoding block	1
Action head	MLP	Action head hidden dim	512
Action head hidden ddepth	2	Action chunking length	10
Optimizer	Adam	Mini-batch size	64

Table 9: **Hyperparameters for Skill Routing Transformer policy.**

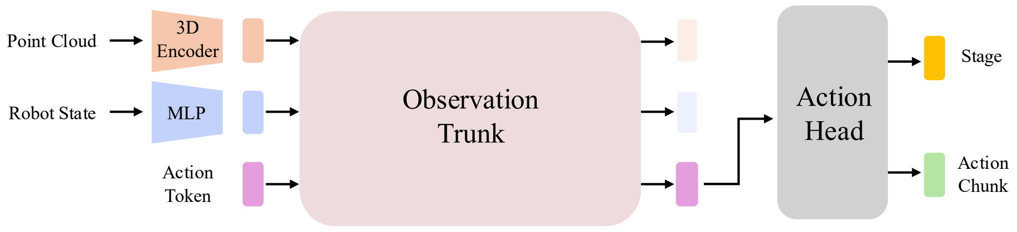

Figure 14: **Skill Routing Transformer Policy Architecture**.

C Experiment Settings and Evaluation Details

C.1 Training Demonstration Dataset

By using the teleoperation system introduced in Sec. A.2, we collect 15 trajectories for each of the three manipulation tasks (Liquid Handling, Plant Watering, and Light Bulb Assembly).

C.2 Experiment Settings

For each task, we use 15 real trajectories to train LODESTAR and all the baselines (Real-only, T-STAR [23], Seq-Dex [25], MimicGen [6], and SkillMimicGen [8]).

We evaluate LODESTAR and all the baselines for 20 trials. We start each trial with different poses for both the robot and the objects in the workspace. We try our best to ensure consistent initial conditions for the evaluation of different methods. For each of the three manipulation tasks, the robot has to sequentially complete the operations defined in Sec. A.3. After completing the operations sequentially, we terminate each trial and call it a *success* when

- in the Liquid Handling task, the tip falls into the container below,
- in the Plant Watering task, the spray bottle is put into the cardboard box,
- in the Light Bulb Assembly task, the light bulb is illuminated.

C.3 Robustness under out-of-distribution (OOD) conditions

We visualize the OOD experiments with a larger initial distribution condition in Fig 15. The red and blue rectangles represent the distribution of demonstration data for the initial pose of the light bulb and the black base. The multiple light bulbs and black bases illustrate different initial positions across evaluation episodes. Faded ones indicate failures, while solid ones indicate successes. Lodestar demonstrates stronger generalizability than the real-only baseline with fewer demonstrations.

C.4 Failure Cases Analysis

We performed a statistical analysis of the failure occurrences of LODESTAR during the main experiments, with the specific stages of failure illustrated in Fig.16. The data indicate that a majority of failures manifested across various skill stages. However, a subset of failures also occurred during the transition stages. Our investigation revealed that these transition stage failures are typically attributable to two primary causes: (1) even in instances of passive contact, minute in-hand displacements occasionally led to the object falling; and (2) occasional collisions where the grasped object, undergoing rapid motion during the transition phase, inadvertently struck another object immediately prior to the intended contact.

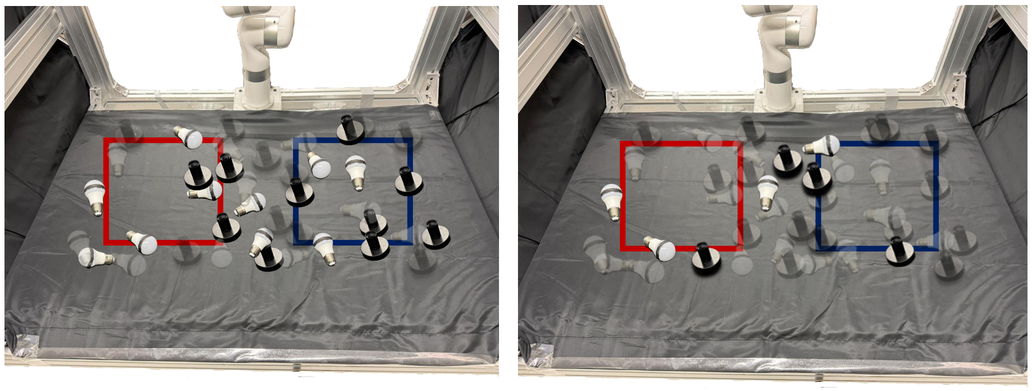

Figure 15: **OOD Evaluation on Larger Init Distributions**.

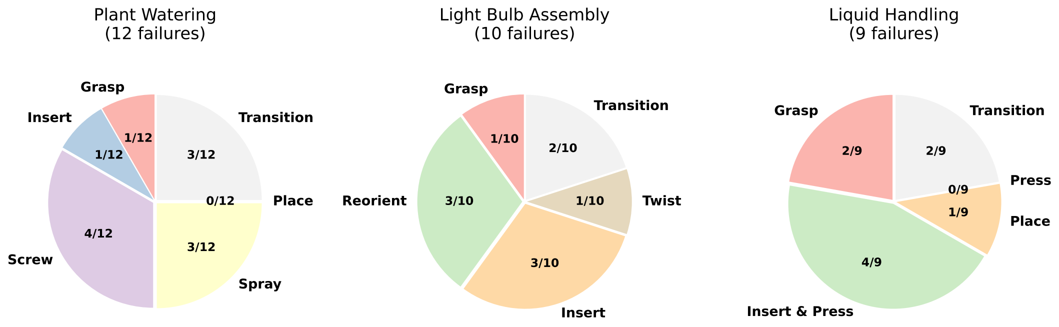

Figure 16: **Failure Cases Distribution on Three Tasks**.