DEX-ONE2MANY: LEARNING DEXTEROUS MANIPULATION FROM A SINGLE HUMAN DEMONSTRATION

Anonymous authors
Paper under double-blind review

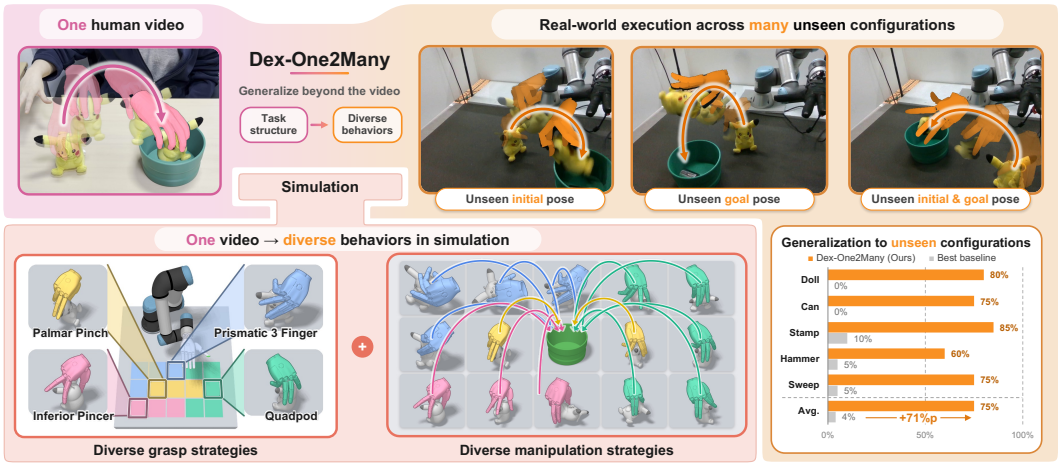


Figure 1: **From one human video to many dexterous behaviors.** Given *one* human video of a manipulation task, Dex-One2Many learns a dexterous policy entirely in simulation and deploys it zero-shot in the real world. By training across diverse simulated task configurations, the resulting policy executes the task across *many* unseen configurations beyond the execution shown in the video.

ABSTRACT

While learning dexterous manipulation from a single human video offers a promising alternative to costly robot demonstrations, many recent methods predominantly imitate demonstrated motions. Such strict motion matching often limits generalization to initial object poses, goal poses, and grasps not shown in the video. Alternatively, discovering a policy via reinforcement learning (RL) allows for broad generalization, but without prior guidance, it struggles with high-dimensional exploration in complex, multi-stage tasks. To address these coupled generalization and exploration challenges, we present **Dex-One2Many**, a real-to-sim-to-real framework that learns a generalizable dexterous manipulation policy from a single human video. Our key insight is to abstract the video into sequential scene graphs that guide RL, enabling efficient exploration while preserving broad generalizability. The graphs serve as generative constraints for sampling diverse reset states and provide dense rewards for each stage. Because the graphs constrain relations rather than exact poses, these reset states cover object poses and grasps beyond the video, while initializing each stage from them with dense rewards keeps exploration short and guided. Trained entirely in simulation, Dex-One2Many transfers zero-shot to a real multi-fingered hand. Across five tool-use and manipulation tasks, Dex-One2Many exceeds baselines by 6.5% in seen configurations, while its robust generalization widens this gap to 71% in unseen scenarios.

1 INTRODUCTION

Dexterous manipulation is a promising direction for enabling robots to perform complex object interactions, but collecting enough high-quality robot demonstrations remains a major bottleneck. For

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

high-degree-of-freedom hands, teleoperation interfaces are particularly costly to design and operate, and the collected motions are tied to a specific robot morphology. Human videos offer a way around this bottleneck: they are easy to collect without a robot or a teleoperation rig and contain rich visual and kinematic information about hand–object interactions. Recent methods exploit this to learn dexterous manipulation from just a single human video (Lum et al., 2025; Paliwal et al., 2026; Han et al., 2026; Zhu et al., 2026).

However, these single-video methods mostly convert the video into execution-level guidance, such as reconstructed object trajectories (Lum et al., 2025), retargeted hand motions (Paliwal et al., 2026; Han et al., 2026), or contact cues (Zhu et al., 2026). Such guidance facilitates high-dimensional dexterous policy learning, but also ties the resulting policy to the motion and spatial configuration shown in the video. Even for the same task, these methods generalize poorly to initial object poses, goal poses, and grasps not shown in the video.

Learning a dexterous policy that generalizes beyond the execution shown in the video raises two tightly coupled challenges. The first is *task abstraction*: deciding what to preserve from the video and what to leave free to vary. Preserving too much reduces the policy to replaying the recorded motion; preserving too little leaves the task underspecified. The second is a *hard exploration problem*: the more that is left free to vary, the more of the state space the policy must explore. Without execution-level guidance, reinforcement learning (RL) struggles to discover multi-finger, contact-rich interactions in this high-dimensional space, especially in multi-stage tasks.

To address both challenges, we present **Dex-One2Many**, a real-to-sim-to-real framework that learns a generalizable dexterous manipulation policy from a single human video (Fig. 1). Our key idea is to extract not the video's motion but its task structure, a stage-wise scene graph specifying which hand–object and object–object relations must hold, and in what order, while leaving RL to learn how the hand acts. To this end, we propose two techniques (Fig. 2).

First, *Stage-Wise Scene Graph Abstraction* reduces the video to the relations the task requires, such as the hand holding an object or an object resting inside a container, and to the order in which they occur. Object poses, goal poses, and grasps remain free to vary. The abstraction splits the video into stages when these relations change and encodes each stage as a graph of predicates over the hand, objects, and their functional parts. Each stage graph describes not one recorded state but a set of states that all preserve the task.

Second, *Scene-Graph-Grounded RL* uses each stage graph as a generative constraint: it samples reset states that satisfy the stage's relations, with diverse multi-finger grasps, and keeps only those that pass our simulation checks. One video thus becomes many training states across the whole workspace. The accepted states form the initial-state distribution of an MDP whose observations and rewards are derived from the graph. Starting episodes at every stage eases exploration: the policy can practice the last stage before mastering the first, rather than reaching it only by chance. Within each stage, predicate-derived rewards give a dense signal of progress toward satisfying the stage's relations, without reference to a demonstrated trajectory. The policy therefore learns the task structure rather than the recorded execution.

Trained entirely in simulation, the resulting policy completes the demonstrated task zero-shot on a real multi-fingered hand across initial and goal object poses not shown in the video. We evaluate five tasks spanning direct object manipulation and tool use on a UR3 arm with a 20-DoF Wuji 1 hand. On configurations with initial and goal poses not shown in the video, Dex-One2Many achieves a 60–85% real-world success rate, while the baselines remain at or below 15%. Our main contributions are as follows:

- **Stage-Wise Scene Graph Abstraction:** We represent a human demonstration as a stage-wise scene graph that retains task-defining structure—the required hand–object and object–object relations and their ordering—while discarding execution-specific motion.
- **Scene-Graph-Grounded RL:** We use each stage's scene graph to generate diverse reset states through the task, enabling the policy to explore alternative grasps and motions while grounding its observations and rewards from the graph.
- **Zero-shot sim-to-real transfer on unseen configurations:** Across five real-world tasks, policies learned from a single human video achieve 75% mean success on configurations not shown in the input video, exceeding the next-best baseline by 71 percentage points.

2 Related Work

Generalizing dexterous policies beyond the execution shown in a single human video raises two key challenges identified in the introduction: *task abstraction* and *hard exploration*. We review the closest work in these two directions, along with broader related work, in Appendix A.

Task Structure Extraction from Human Videos. Human videos have been used to learn reusable visual representations (Nair et al., 2022; Lee et al., 2026c), rewards (Ma et al., 2023; Kumar et al., 2022; Huang et al., 2024a), point tracks (Bharadhwaj et al., 2024), latent plans (Wang et al., 2023a), or action priors (Shaw et al., 2022) from large video collections. In the single-video setting, however, generalization depends largely on how the video is abstracted. Prior work abstracts the video into keypoints, keyframes, object-centric graphs, or object pose trajectories, then adapts the demonstrated execution to a new scene by warping or tracking it, or by synthesizing imitation data around it (Gao et al., 2023; Sieb et al., 2020; Heppert et al., 2024; Zhu et al., 2024; Li et al., 2024; Zhou et al., 2025; Tang et al., 2025; Heppert et al., 2026). Because what transfers is the demonstrated motion, adaptation extends only as far as that motion can be reshaped. Relational task specifications provide a more abstract alternative and are well established in symbolic and scene-graph planning (Fikes & Nilsson, 1971; Rana et al., 2023), and vision-language models have extracted symbolic task plans from a human video (Wake et al., 2024) and stage-wise relational constraints from language (Huang et al., 2024b). Building on this relational view, our *Stage-Wise Scene Graph Abstraction* extracts task structure from the single video, while discarding execution details, encoding each stage as a graph whose edges are predicates stating which hand–object and object–object relations must hold, in the order the video shows. Unlike ORION's graphs (Zhu et al., 2024), our stage-wise scene graph carries no coordinates or keypoint trajectories from the video, so each graph describes not one state but the set of all states that satisfy its predicates. Rather than a plan to execute, a reference to adapt, or a cost to optimize, each graph serves as a constraint that sampled training states must satisfy, leaving object poses, goal poses, and grasps free to vary. RL then learns how the hand acts instead of inheriting the demonstrated motion.

Learning Dexterous Manipulation from a Single Human Video. Dexterous manipulation poses a hard exploration problem for RL because multi-fingered hands must coordinate many joints through contact-rich interactions (Rajeswaran et al., 2018). Prior methods that learn dexterous manipulation from human video therefore extract execution-level guidance, such as object pose trajectories used as a tracking reward (Lum et al., 2025; Zhu et al., 2026; Sharma et al., 2026; Chen et al., 2025b; Mandi et al., 2026), retargeted hand motions used as references to imitate or refine (Qin et al., 2022; Li et al., 2025; Pan et al., 2025; Paliwal et al., 2026; Han et al., 2026; Guzey et al., 2025), or demonstrated contacts (Zhu et al., 2026; Sharma et al., 2026). This guidance makes exploration tractable but ties the resulting behavior to the demonstrated motion, even when another initial object pose, goal pose, or grasp requires a different execution of the same task. As a result, their generalization is largely limited to local variations around the demonstrated configuration (Lum et al., 2025; Han et al., 2026; Zhu et al., 2026).[1] Removing execution-level guidance, however, creates the second challenge: how to make RL explore efficiently from task structure alone. Reset-based RL alleviates hard exploration by starting episodes near a known goal (Florensa et al., 2017) or along a demonstration (Resnick et al., 2018; Salimans & Chen, 2018; Tao et al., 2024; Bauza et al., 2025), so the policy learns from later states before it can reach them. Demonstration-based resets, however, inherit the demonstrated execution. More recent approaches such as OmniReset (Yin et al., 2026) and Video2Policy (Ye et al., 2025) reduce this dependence by generating reset states or task programs, but rely on user-specified task information or video-derived object states and have only been demonstrated with parallel-jaw grippers. Our approach grounds resets, observations, and rewards in a stage-wise scene graph derived from the human video, so it derives the task structure without imitating the execution.

3 Method

Given a single demonstration of a manipulation task, our goal is to learn a dexterous policy that generalizes to diverse spatial configurations, including randomized object poses and novel grasps, beyond the original video. To achieve this, we first abstract the demonstration into a stage-wise scene graph (Sec. 3.1). We then formulate policy learning as a Markov decision process (MDP)

[1] Human2Sim2Robot: initial position within $4\,\text{cm} \times 4\,\text{cm}$, fixed goal; Video2Sim2Real: initial/goal within $\pm 5\,\text{cm}$, $\pm 10^\circ$.

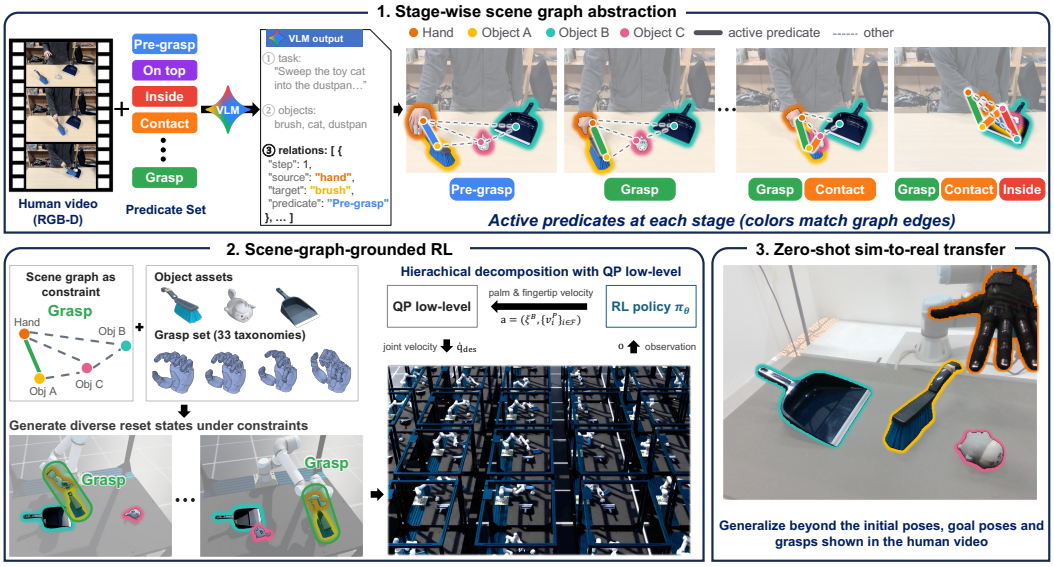


Figure 2: **Overview of Dex-One2Many.** (1) From a single human video, a VLM extracts a stage-wise scene graph of object relations. (2) Each stage graph constrains the sampling of object poses and synthesized grasps, yielding verified reset states for every stage. An RL policy trained from these resets with predicate-derived rewards acts in task space through a QP controller. (3) The policy transfers zero-shot to unseen initial poses, goal poses, and grasps on real hardware.

grounded in this graph, with observations, rewards, and an initial-state distribution constructed from it without task-specific design (Sec. 3.2). The learned policy then transfers zero-shot to the real robot (Sec. 3.3). Fig. 2 summarizes the full pipeline.

3.1 Task Abstraction as a Stage-wise Scene Graph

We abstract the human video as a sequence of stage-wise scene graphs that preserve which relations must hold and in what order, while allowing object poses, goal poses, grasps, and trajectories to vary. Each graph represents one stage of the task: its nodes identify the hand, task-relevant objects, and their functional parts, while its edges describe the relations characterizing that stage. When the same task is performed from different configurations, the poses and trajectories may differ, but the required relations and their order remain the same. We refer to this abstraction as *Stage-Wise Scene Graph Abstraction*.

To encode the relations carried by the scene-graph edges without fixing the poses observed in the video, we use a predefined, task-independent *predicate* vocabulary (Table 3). Some predicates specify relations that must hold, such as `grasp`, `contact`, and `inside`, while others describe proximity toward the next relation to be established, such as `pre_grasp` and `pre_inside`. The former describe what must hold in the current stage, whereas the latter indicate which relation should be established next. Together, these predicates describe task progression without fixing the specific configuration shown in the video.

We recover these stage-wise graphs from the video using a vision-language model (VLM) that abstracts the task directly. Given the video and the predicate vocabulary, the VLM identifies task-relevant entities and functional parts as graph nodes and infers the predicate relations between them over time. We introduce a new stage whenever the predicate relations change, either because a new relation is added or because a relation being approached becomes established, such as `pre_grasp` transitioning to `grasp`. This yields an ordered sequence $\mathcal{G}_{1:K} = (\mathcal{G}_1, \ldots, \mathcal{G}_K)$. Fig. 2 illustrates how the stage-wise scene graph captures task progression in the sweeping task: a new stage is introduced when `pre_grasp` becomes `grasp`; in later stages, the established `grasp` and `contact` relations remain as constraints, while the toy cat is moved toward the dustpan until the new `inside` relation is established.

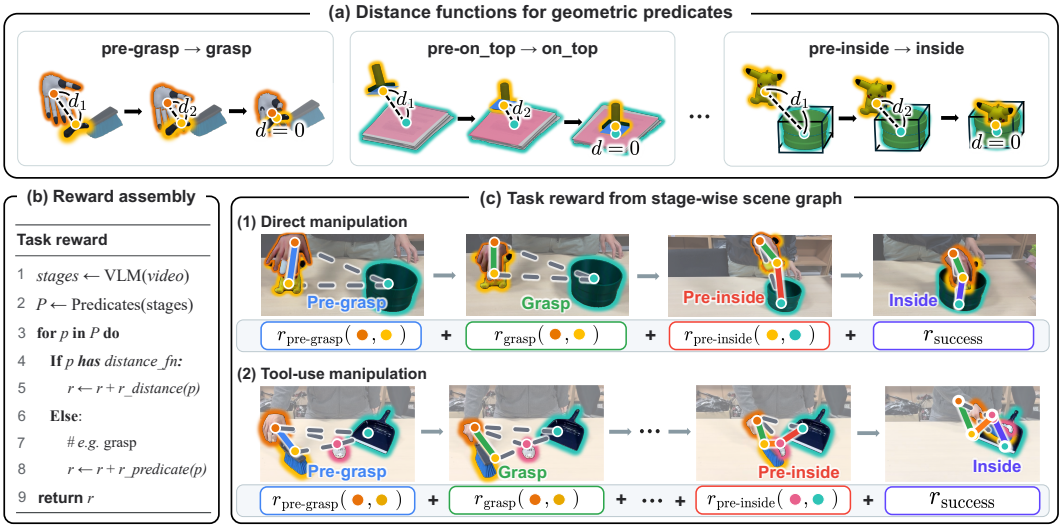


Figure 3: **Using scene-graph predicates for policy learning.** (a) Predicates that can be expressed geometrically are mapped to Cartesian-space distance functions. (b) The predicates in a stage-wise scene graph determine which reward terms are assembled into the task reward. (c) The same assembly rule is instantiated for direct and tool-use manipulation from their respective scene graphs.

3.2 Scene-graph-grounded Reinforcement Learning

Given the scene-graph sequence $\mathcal{G}_{1:K}$ from Sec. 3.1, we formulate policy learning as a finite-horizon MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, p, r, \gamma, \rho_0, T \rangle$. Here, γ and T denote the discount factor and episode horizon; the simulator provides transition dynamics p, and $\mathcal{A}$ consists of desired palm and fingertip velocities tracked by a low-level QP controller (Sec. 3.3). The scene graph defines the policy observation o from $\mathcal{S}$, the reward r, and the initial-state distribution ρ_0, as described below.

Observation from the scene graph. We construct the policy observation o from the task-relevant state associated with the scene graph's nodes and edges. Nodes are represented by role: the hand node provides robot proprioception, object nodes encode object geometry and 6D pose, and functional-part nodes localize regions such as the grasp region. Edges capture the relative 6D pose between connected entities. Because these encodings are shared across tasks, the same observation construction applies to different task graphs. Concatenating these node and edge features forms the policy observation, making the observation design reusable (see Appendix C.1 for details).

Task reward from the scene graph. Each scene-graph edge encodes a predicate between two entities, which we map to a geometric or physical reward. Geometric predicates use the distance metric in Table 3 as a dense progress signal, a 3D Cartesian distance for spatial relations (e.g., to a grasp region for `pre_grasp` or a receptive region for `pre_inside`; Fig. 3(a)) and a joint-space distance for `seat`. For predicates not naturally represented by such a distance, we use a predicate-specific physical measure. Specifically, we evaluate `grasp` using CHORD's force-closure reward in 6D wrench space (Zhu et al., 2026), augmented with a grasp-quality measure (Lynch & Park, 2017) to ensure grasp stability (see Appendix C.2 for details and discussion).

With a reward defined for each predicate, we construct the task reward from the stage-wise scene graph. As the task progresses to a new stage, the reward for the newly introduced predicate is added to the task reward (Fig. 3(b)). Because each predicate has a fixed reward definition, we apply the same construction across different tasks. Fig. 3(c) shows this shared construction for direct and tool-use manipulation tasks: although they involve different entities and predicate sequences, their task rewards are assembled from the same predicate-level reward functions.

Initial-state distribution from the stage-wise scene graphs. While predicate rewards guide progress from a state, the stage-wise scene graph defines the initial-state distribution by determining which parts of the task the policy can practice during training. Since each stage graph $\mathcal{G}_k$ fixes relations but not poses or grasps, we construct ρ_0 by using it as a generative constraint. For geometric predicates, the same feature distance that provides their dense progress signal also serves as a placement rule: we position the source node relative to its target node so that the condition in Table 3

holds, applying these rules along the edges of $\mathcal{G}_k$ (Fig. 10, Appendix C.3). For the final stage, we first sample a state that satisfies the goal predicate and then perturb it into a near-goal state, since an already successful reset provides less useful learning signal. Episodes start uniformly from such resets across all stages, so the policy practices late interactions without first reaching them from the task start, which reduces the exploration burden of long-horizon tasks (Yin et al., 2026).

For `grasp`, the placement rule is grasp synthesis. Instead of fixing a single grasp, we place the hand at one of many multi-finger grasps generated by Dexonomy (Chen et al., 2025a) within the object's grasp region. The policy thus encounters diverse hand–object configurations during training(Appendix C.3.2). Each grasp also includes the finger command q_{cmd} that closes the hand on the object, which we restore at initialization so the grasp holds from the first step. Before a sampled state enters ρ_0, we instantiate it in simulation and keep it only if it passes our checks for scene collision and reset stability, using grasps certified beforehand under external forces (Appendix C.3.3).

3.3 Zero-shot Sim-to-Real Transfer

We train the policy with PPO (Schulman et al., 2017) using an asymmetric actor–critic (Pinto et al., 2018), in which the critic receives privileged observations and the actor receives only the scene-graph-grounded observation available on the real robot. To enable zero-shot transfer, we randomize physical properties such as object mass and friction during training (Appendix C.5) and address the sim-to-real gap in perception and control. On the perception side, we perturb the actor observations during training with object-pose noise, perceived-size noise, and observation delay. At deployment, we estimate object poses from two RGB-D cameras using FoundationPose++ (Yan & Chu, 2025) and fuse the estimates with depth-consistency weights that down-weight unreliable views (Appendix D). On the control side, the policy outputs desired palm and fingertip velocities, which a QP low-level controller (Lee et al., 2026a) maps to joint velocities while enforcing kinematic and deployment-time safety constraints. This lets us adjust joint-velocity limits and add collision-avoidance constraints at deployment without retraining the policy. Appendix C.4 provides further details on the QP controller and demonstrates a real-world hardware failure prevented by the QP constraints.

4 Experiments

In this section, we evaluate Dex-One2Many in simulation and on real hardware. Our experiments address three questions:

- **(Q1)** Does Dex-One2Many generalize to initial and goal configurations not shown in the single human video?
- **(Q2)** Does Dex-One2Many apply across diverse dexterous embodiments with different kinematics?
- **(Q3)** Which design choices of Dex-One2Many are critical to its performance?

4.1 Experimental Setup

Tasks. We evaluate Dex-One2Many on five manipulation tasks that span direct object manipulation and tool use (Fig. 4). The direct-manipulation tasks (Doll, Can, Stamp) require the robot to grasp an object and move it to a target configuration, while the tool-use tasks (Hammer, Sweep) require the robot to grasp and manipulate a tool to interact with another object.

Hardware Setup. We conduct all real-world experiments on a 6-DoF UR3 robot arm equipped with a 20-DoF Wuji 1 dexterous hand. Two Intel RealSense D435i RGB-D cameras capture the workspace from complementary viewpoints. We estimate object poses from these observations using FoundationPose++ (Yan & Chu, 2025).

Baselines. We compare our Dex-One2Many against two prior methods that learn dexterous manipulation from a single human video. These methods differ in what information they extract from the human demonstration and how they use it for policy learning. Do as I Do (Paliwal et al., 2026) reconstructs the 4D hand–object interaction and retargets the recovered human hand motion to the robot through trajectory optimization. CHORD (Zhu et al., 2026) uses the demonstrated object-state evolution as an imitation reward and augments it with contact-wrench guidance that encourages the robot to reproduce the demonstrated trajectories.

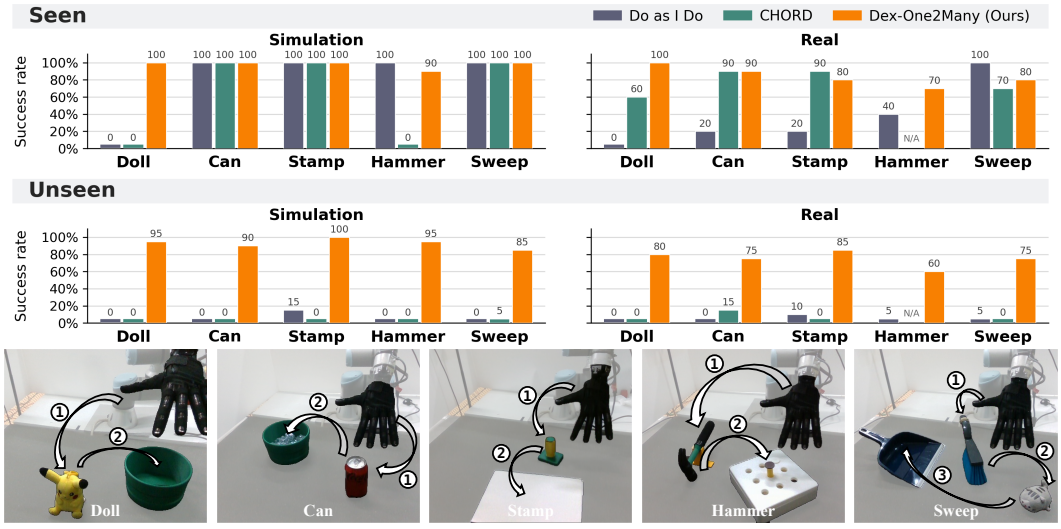


Figure 4: **Generalization beyond the human video in simulation and the real world.** Success rates are reported for configurations shown in the input human video (*Seen*, top) and configurations with different initial and goal poses not shown in the video (*Unseen*, bottom), in both simulation and the real world. The bottom row illustrates the five evaluated tasks; N/A denotes an experiment unavailable due to a hardware failure.

4.2 Does Dex-One2Many Generalize Beyond the Human Video?

Evaluation protocol. We evaluate generalization in both simulation and the real world using the same *Seen*/*Unseen* protocol. For each method and task, we evaluate 30 episodes in simulation and 30 trials on the real robot: 10 in *Seen* configurations and 20 in *Unseen* configurations. The *Seen* configurations reproduce the initial and goal poses shown in the input human video, whereas the *Unseen* configurations use different initial and goal poses not present in the video. All evaluation episodes start with an empty hand and require the policy to execute the full task from reaching to task completion. We deploy real-world policies zero-shot, without real-world data or fine-tuning.

Results. Figure 4 compares performance on *Seen* and *Unseen* configurations in simulation and the real world. On *Seen* configurations, the baselines can achieve high success on several tasks, showing that execution-level guidance extracted from the human video can reproduce the demonstrated task configuration. The difference becomes pronounced on *Unseen* configurations. Across the evaluated baselines, success remains at or below 15% in both simulation and the real world, whereas Dex-One2Many achieves 85–100% success in simulation and 60–85% on the real robot.

Figure 5 further isolates generalization to unseen initial object positions on the Can task. With the goal fixed, the baselines succeed primarily near the demonstrated object position, whereas Dex-One2Many maintains high success across a substantially broader range of the workspace. This shows that the performance gap is not limited to a particular set of evaluation configurations, but reflects broader spatial generalization beyond the demonstrated execution. Additional results across the remaining tasks are provided in Appendix E.1.

The baselines derive execution-level guidance from the demonstrated hand or object motion, which is tied to the execution observed in the video. In contrast, Dex-One2Many uses stage-wise scene graphs as generative constraints for diverse task-consistent training states, allowing the policy to learn the same task structure across different configurations beyond the demonstrated execution.

4.3 Does Dex-One2Many Apply Across Diverse Dexterous Embodiments?

Evaluation protocol. We evaluate Dex-One2Many in simulation across four arm–hand embodiments with different kinematics: UR3 + Wuji 1, UR5e + Sharpa Wave, UR5e + Wuji 2, and UR5e + Allegro. For each embodiment, we only replace the robot URDF and synthesize a corresponding grasp set, keeping the task specification and all other components unchanged. We train a separate

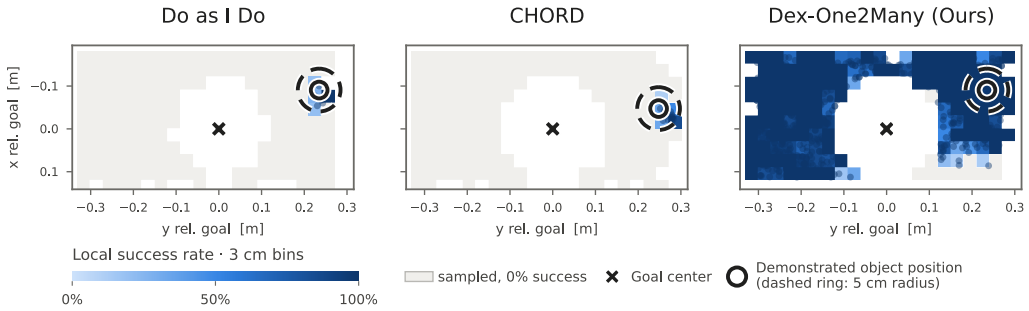


Figure 5: **Generalization to unseen initial object positions on the Can task.** In simulation with the goal fixed, Dex-One2Many succeeds across a broader range of initial positions than the baselines.

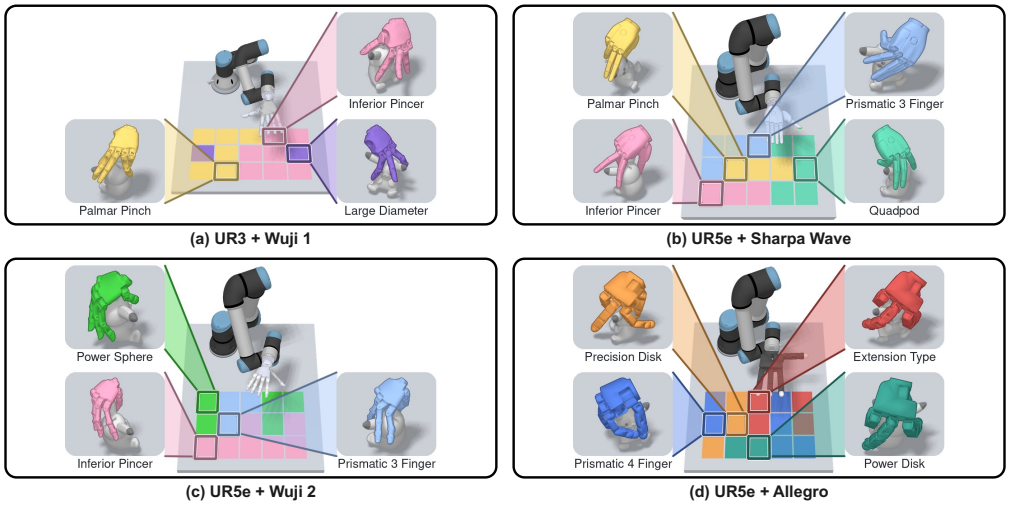


Figure 6: **Diverse grasp strategies across dexterous embodiments and object configurations on the Doll task.** Across the 33-type grasp taxonomy (Feix et al., 2015), different embodiments learn different grasp types, and the same embodiment adapts its grasp across object configurations.

policy for each embodiment and report success rates over 500 evaluation episodes per task, with initial object positions sampled within the reachable part of $(x, y) \in [0.3, 0.7] \times [-0.17, 0.4]$ m.

Results. Table 1 reports task-wise success rates and the mean across five tasks for four distinct dexterous embodiments. Despite substantial differences in arm–hand kinematics, Dex-One2Many achieves consistently high performance across all four tested embodiments, with mean success rates ranging from 94.6% to 96.8%. Appendix E.3 provides rollouts across all five tasks and multiple configurations. Figure 6 further shows that the learned execution is embodiment-specific. Under the same task specification, different hands adopt different grasp types, and the same hand can change its grasp as the object configuration changes. Rather than retargeting a fixed human grasp, Dex-One2Many exposes RL to diverse grasps synthesized for each morphology, and the policy learns which grasp suits each scene configuration. Appendix E.2 provides additional grasp examples across embodiments and configurations.

Table 1: **Simulation performance across diverse dexterous embodiments.** Success rates (%) over 500 episodes per task.

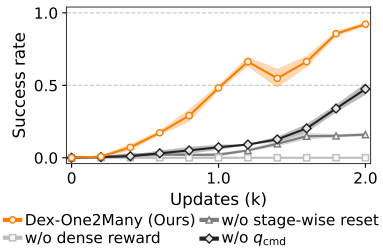
	UR3 Wuji 1	UR5e Sharpa	UR5e Wuji 2	UR5e Allegro
Doll	88.4	98.4	97.2	97.2
Can	93.6	97.4	95.2	92.2
Stamp	98.6	96.6	97.0	98.8
Hammer	94.6	94.2	97.6	91.8
Sweep	98.0	96.6	97.0	100.0
Mean	94.6	96.6	96.8	96.0

4.4 Which Design Choices Are Critical to Dex-One2Many?

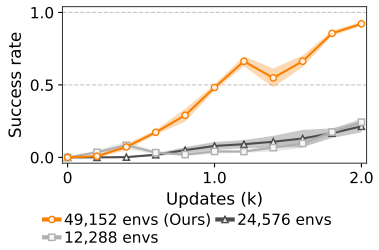
Evaluation protocol. We conduct all ablation studies on the Sweep task in simulation. Success is measured on full-task episodes from empty-hand starts. We study two aspects of Dex-One2Many: the contribution of its learning components and the effect of training scale. For component ablations, we remove one component at a time while keeping all other training settings unchanged. For the training-scale analysis, we vary only the number of parallel environments used for PPO training.

Component ablations. From the human video, Dex-One2Many extracts a stage-wise scene graph that specifies the task relations at each stage. We examine three design choices in how this task structure is used for policy learning. **(1)** Removing stage-wise resets (w/o stage-wise reset) drops success after 2k PPO updates from over 90% to about 16%, showing that starting from intermediate stages is key to exploring multi-stage tasks. **(2)** Removing predicate-derived rewards (w/o dense reward) keeps the success rate at 0% throughout training, showing that dense feedback toward the required task relations is critical for guiding exploration. **(3)** Finally, omitting the verified closing command (w/o q_{cmd}) lowers success to about 48% after 2K PPO updates and induces frequent grasp slips and drops, showing that restoring active contact forces is vital for stable transitions across stages.

Effect of training scale. Increasing the number of parallel environments substantially improves learning speed and final performance. With 49,152 environments, Dex-One2Many reaches over 90% success, whereas training with 24,576 or 12,288 environments remains below 25% within the same number of PPO updates. These results show that large-scale parallel experience collection substantially benefits policy learning under our diverse reset distribution.



(a) Component ablations



(b) Training scale

Figure 7: **Ablation studies.** (a) Removing each component. (b) Varying the number of environments.

5 Conclusion

We presented Dex-One2Many, a real-to-sim-to-real framework for learning generalizable dexterous manipulation from a single human video. Rather than treating the demonstrated motion as an execution reference, Dex-One2Many abstracts the video into a stage-wise scene graph that preserves the task structure while leaving object poses, goal poses, grasps, and motion paths free to vary. Scene-Graph-Grounded RL then uses this structure to construct diverse stage-wise reset states and predicate-derived rewards, making exploration tractable without inheriting the demonstrated trajectory. Across five manipulation tasks, the resulting policies generalize to configurations not shown in the video and transfer zero-shot to a real multi-fingered hand. These results suggest that extracting *what* must be achieved from a human video, while allowing RL to discover *how* to achieve it, provides a promising route from a single demonstration to diverse dexterous behaviors.

Limitations and future work. Although Dex-One2Many substantially improves generalization across unseen configurations and applies to diverse hand morphologies, several limitations remain. First, a sim-to-real gap remains because real-world deployment relies on estimated 6D object poses. Pose-estimation noise and differences between simulated and real object geometry can degrade policy performance, particularly during contact-rich interactions. Second, our real-to-sim pipeline reconstructs only rigid objects, so articulated objects would require manually built assets. Since a single video may not reveal the underlying articulation structure, such as joint locations, axes, and joint limits, automatic real-to-sim reconstruction remains difficult. Inferring these structural kinematics directly from unconstrained video is an important avenue for future work.

AI USE STATEMENT

In this work, we used generative AI tools to assist with polishing the writing, including translating author-written drafts into English. We have not used generative AI tools for developing theoretical models, mathematical proofs, or methodology. Additionally, we used generative AI tools for grammar checking to improve readability. We have reviewed all AI-assisted work. For example, we checked all edited text to ensure it preserves the original meaning and technical content. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not include human subjects, and no personally identifiable information was collected or used.

REPRODUCIBILITY STATEMENT

We provide the implementation and evaluation details needed to reproduce our results throughout the paper and appendix. The proposed method is described in Section 3.1–3.3, and the experimental setup and evaluation protocols are provided in Section 4. Appendix C.5 specifies the simulation setup, policy architecture, PPO hyperparameters, computational setup, and domain-randomization parameters. Appendix C.4 provides the formulation of the QP low-level controller and its deployment-time constraints. Additional appendices describe the reset-state and grasp construction procedures.

REFERENCES

Ananye Agarwal, Shagun Uppal, Kenneth Shaw, and Deepak Pathak. Dexterous functional grasping. In *Conference on Robot Learning (CoRL)*, 2023.

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement-residual rl for precise assembly. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

Seung Hyeon Bang, Carlos Arribalzaga Jové, and Luis Sentis. Rl-augmented mpc framework for agile and robust bipedal footstep locomotion planning and control. In *IEEE-RAS 23rd International Conference on Humanoid Robots (Humanoids)*, 2024.

Maria Bauza, Jose Enrique Chen, Valentin Dalibard, Nimrod Gileadi, Roland Hafner, Murilo F. Martins, Joss Moore, Rugile Pevceviciute, Antoine Laurens, Dushyant Rao, Martina Zambelli, Martin Riedmiller, Jon Scholz, Konstantinos Bousmalis, Francesco Nori, and Nicolas Heess. Demostart: Demonstration-led auto-curriculum applied to sim-to-real with multi-fingered robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

Homanga Bharadhwaj, Roozbeh Mottaghi, Abhinav Gupta, and Shubham Tulsiani. Track2act: Predicting point tracks from internet videos enables generalizable robot manipulation. In *European Conference on Computer Vision (ECCV)*, 2024.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. Sam 3: Segment anything with concepts. In *International Conference on Learning Representations (ICLR)*, 2026.

Jiayi Chen, Yuxing Chen, Jialiang Zhang, and He Wang. Task-oriented dexterous hand pose synthesis using differentiable grasp wrench boundary estimator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

Jiayi Chen, Yubin Ke, Lin Peng, and He Wang. Dexonomy: Synthesizing all dexterous grasp types in a grasp taxonomy. In *Robotics: Science and Systems (RSS)*, 2025a.

Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, et al. Sam 3d: 3dfy anything in images. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026a.

Yifei Chen, Shihan Lu, and Colgate Kevin Lynch. Robust in-hand manipulation via priors in reinforcement learning and mechanical design. *arXiv preprint arXiv:2607.12105*, 2026b.

Zerui Chen, Shizhe Chen, Etienne Arlaud, Ivan Laptev, and Cordelia Schmid. ViViDex: Learning vision-based dexterous manipulation from human videos. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2025b.

Prithwish Dan, Kushal Kedia, Angela Chao, Edward Weiyi Duan, Maximus Adrian Pace, Wei-Chiu Ma, and Sanjiban Choudhury. X-sim: Cross-embodiment learning via real-to-sim-to-real. In *Conference on Robot Learning (CoRL)*, 2025.

Thomas Feix, Javier Romero, Heinz-Bodo Schmiedmayer, Aaron M Dollar, and Danica Kragic. The grasp taxonomy of human grasp types. *IEEE Transactions on human-machine systems*, 46 (1):66–77, 2015.

Carlo Ferrari and John Canny. Planning optimal grasps. In *IEEE International Conference on Robotics and Automation (ICRA)*, 1992.

Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2017.

Siddhant Gangapurwala, Mathieu Geisert, Romeo Orsolino, Maurice Fallon, and Ioannis Havoutis. Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control. *IEEE Transactions on Robotics*, 38(5):2908–2927, 2022.

Jianfeng Gao, Zhi Tao, Noémie Jaquier, and Tamim Asfour. K-vil: Keypoints-based visual imitation learning. *IEEE Transactions on Robotics*, 39(5):3888–3908, 2023.

Gemini Team, Google. Gemini 3.1 pro: A smarter model for your most complex tasks. `https://blog.google/innovation-and-ai/models-and-research/gemini-models/gemini-3-1-pro/`, 2026. Accessed: 2026-09-26.

Irmak Guzey, Yinlong Dai, Georgy Savva, Raunaq Bhirangi, and Lerrel Pinto. Bridging the human to robot dexterity gap through object-oriented rewards. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

Yunhai Han, Jianuo Qiu, Linhao Bai, Ziyu Xiao, Zihang Zeng, Yangcen Liu, Zhaodong Yang, Shalin Jain, Wenrui Ma, Jiaqi Fu, Yuqian Zheng, Manisha Natarajan, Muhammad Zubair Irshad, Kenneth Shaw, Matthew Gombolay, Zsolt Kira, and Harish Ravichandar. Video2sim2real: Full-stack autonomous dexterous skill acquisition from a single human video. *arXiv preprint arXiv:2606.08828*, 2026.

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

Nick Heppert, Max Argus, Tim Welschehold, Thomas Brox, and Abhinav Valada. Ditto: Demonstration imitation by trajectory transformation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

Nick Heppert, Minh Quang Nguyen, and Abhinav Valada. Scaling single human demonstrations for imitation learning using generative foundational models. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026.

Lei Huang, Weijia Cai, Zihan Zhu, Chen Feng, Helge Rhodin, and Zhengbo Zou. Virl: Self-supervised visual graph inverse reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2024a.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. In *Conference on Robot Learning (CoRL)*, 2024b.

Kushal Kedia, Tyler Ga Wei Lum, Jeannette Bohg, and C. Karen Liu. Simtoolreal: An object-centric policy for zero-shot dexterous tool manipulation. In *Robotics: Science and Systems (RSS)*, 2026.

Sateesh Kumar, Jonathan Zamora, Nicklas Hansen, Rishabh Jangir, and Xiaolong Wang. Graph inverse reinforcement learning from diverse videos. In *Conference on Robot Learning (CoRL)*, 2022.

Ho Jae Lee, Yonghyeon Lee, Alexander Alexiev, Tzu-Yuan Lin, Se Hwan Jeon, and Sangbae Kim. Learning reactive dexterous grasping via hierarchical task-space rl planning and joint-space qp control. *arXiv preprint arXiv:2605.03363*, 2026a.

Jayjun Lee, Jessica Yin, Asif Rana, Nicholas Blauch, Sam Mady, Mohak Bhardwaj, Nima Fazeli, Nathan Ratliff, Karl Van Wyk, and Ankur Handa. Adept: Accelerating dexterity via pre-training and post-training using reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2026b.

Jusuk Lee, Seungjae Lee, Jonghun Shin, Hoseong Jung, Sungha Kim, Daesol Cho, H Jin Kim, Jia-Bin Huang, and Furong Huang. Dynaflip: Rethinking robotics perception via tri-modal-dynamics guided representation. *arXiv preprint arXiv:2605.30350*, 2026c.

Jinhan Li, Yifeng Zhu, Yuqi Xie, Zhenyu Jiang, Mingyo Seo, Georgios Pavlakos, and Yuke Zhu. Okami: Teaching humanoid robots manipulation skills through single video imitation. In *Conference on Robot Learning (CoRL)*, 2024.

Kailin Li, Puhao Li, Tengyu Liu, Yuyang Li, and Siyuan Huang. Maniptrans: Efficient dexterous bimanual manipulation transfer via residual learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Tengyu Liu, Zeyu Liu, Ziyuan Jiao, Yixin Zhu, and Song-Chun Zhu. Synthesizing diverse and physically stable grasps with arbitrary hand structures using differentiable force closure estimator. *IEEE Robotics and Automation Letters*, 7(1):470–477, 2021.

Tyler Ga Wei Lum, Olivia Y. Lee, C. Karen Liu, and Jeannette Bohg. Crossing the human-robot embodiment gap with sim-to-real rl using one human demonstration. In *Conference on Robot Learning (CoRL)*, 2025.

Tyler Ga Wei Lum, Kushal Kedia, C. Karen Liu, and Jeannette Bohg. Play2perfect: What matters in dexterous play pretraining for precise assembly? In *Conference on Robot Learning (CoRL)*, 2026.

Kevin M Lynch and Frank C Park. Modern robotics. *Mechanics, Planning, and Control*, 2017.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. In *International Conference on Learning Representations (ICLR)*, 2023.

Zhao Mandi, Yifan Hou, Dieter Fox, Yashraj Narang, Ajay Mandlekar, and Shuran Song. Dex-machina: Functional retargeting for bimanual dexterous manipulation. In *International Conference on Machine Learning (ICML)*, 2026.

Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Munoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, et al. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.

Ken Museth, Jeff Lait, John Johanson, Jeff Budsberg, Ron Henderson, Mihai Alden, Peter Cucka, David Hill, and Andrew Pearce. Openvdb: an open-source data structure and toolkit for high-resolution volumes. In *ACM SIGGRAPH 2013 Courses*, SIGGRAPH '13, 2013.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2022.

Bhawna Paliwal, Haritheja Etukuru, William Liang, Pieter Abbeel, Nur Muhammad Mahi Shafiullah, and Jitendra Malik. Do as i do: Dexterous manipulation data from everyday human videos. *arXiv preprint arXiv:2606.19333*, 2026.

Chaoyi Pan, Changhao Wang, Haozhi Qi, Zixi Liu, Homanga Bharadhwaj, Akash Sharma, Tingfan Wu, Guanya Shi, Jitendra Malik, and Francois Hogan. Spider: Scalable physics-informed dexterous retargeting. *arXiv preprint arXiv:2511.09484*, 2025.

Juhan Park, Taerim Yoon, Seungmin Kim, Joong-Gil Kim, Wontae Ye, Jeongeun Park, Yoonbyung Chai, Geonwoo Cho, Geunwoo Cho, Dohyeong Kim, Kyungjae Lee, Yong-Jae Kim, and Sungjoon Choi. Learning dexterous grasping from sparse taxonomy guidance. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2026.

Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. In *Robotics: Science and Systems (RSS)*, 2018.

Yuzhe Qin, Yueh-Hua Wu, Shaowei Liu, Hanwen Jiang, Ruihan Yang, Yang Fu, and Xiaolong Wang. DexMV: Imitation learning for dexterous manipulation from human videos. In *European Conference on Computer Vision (ECCV)*, 2022.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Robotics: Science and Systems (RSS)*, 2018.

Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, Ian Reid, and Niko Suenderhauf. Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning. In *Conference on Robot Learning (CoRL)*, 2023.

Cinjon Resnick, Roberta Raileanu, Sanyam Kapoor, Alexander Peysakhovich, Kyunghyun Cho, and Joan Bruna. Backplay: "man muss immer umkehren". *arXiv preprint arXiv:1807.06919*, 2018.

Tim Salimans and Richard Chen. Learning montezuma's revenge from a single demonstration. *arXiv preprint arXiv:1812.03381*, 2018.

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

Satvik Sharma, Samrat Sahoo, Huang Huang, Fei-Fei Li Jiajun Wu, Dorsa Sadigh, and Jeannette Bohg. One demonstration, many objects: Generalizing manipulation via local contact geometry. *arXiv preprint arXiv:2609.01938*, 2026.

Kenneth Shaw, Shikhar Bahl, and Deepak Pathak. Videodex: Learning dexterity from internet videos. In *Conference on Robot Learning (CoRL)*, 2022.

Maximilian Sieb, Zhou Xian, Audrey Huang, Oliver Kroemer, and Katerina Fragkiadaki. Graph-structured visual imitation. In *Conference on Robot Learning (CoRL)*, 2020.

Aaditya Singh et al. OpenAI GPT-5 system card, 2025. URL `https://arxiv.org/abs/2601.03267`.

Chao Tang, Anxing Xiao, Yuhong Deng, Tianrun Hu, Wenlong Dong, Hanbo Zhang, David Hsu, and Hong Zhang. Mimicfunc: Imitating tool manipulation from a single human video via functional correspondence. In *Conference on Robot Learning (CoRL)*, 2025.

Stone Tao, Arth Shukla, Tse kai Chan, and Hao Su. Reverse forward curriculum learning for extreme sample and demonstration efficiency in reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2024.

Marcel Torne, Anthony Simeonov, Zechu Li, April Chan, Tao Chen, Abhishek Gupta, and Pulkit Agrawal. Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation. In *Robotics: Science and Systems (RSS)*, 2024.

Sébastien Valette and Jean-Marc Chassery. Approximated centroidal voronoi diagrams for uniform polygonal mesh coarsening. *Computer Graphics Forum*, 23(3):381–389, 2004.

Naoki Wake, Atsushi Kanehira, Kazuhiro Sasabuchi, Jun Takamatsu, and Katsushi Ikeuchi. Gpt-4v (ision) for robotics: Multimodal task planning from human demonstration. *IEEE Robotics and Automation Letters*, 9(11):10567–10574, 2024.

Weikang Wan, Haoran Geng, Yun Liu, Zikang Shan, Yaodong Yang, Li Yi, and He Wang. Unidexgrasp++: Improving dexterous grasping policy learning via geometry-aware curriculum and iterative generalist-specialist learning. In *International Conference on Computer Vision (ICCV)*, 2023.

Chen Wang, Linxi Fan, Jiankai Sun, Ruohan Zhang, Li Fei-Fei, Danfei Xu, Yuke Zhu, and Anima Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play. In *Conference on Robot Learning (CoRL)*, 2023a.

Ruicheng Wang, Jialiang Zhang, Jiayi Chen, Yinzhen Xu, Puhao Li, Tengyu Liu, and He Wang. Dexgraspnet: A large-scale robotic dexterous grasp dataset for general objects based on simulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023b.

Xinyue Wei, Minghua Liu, Zhan Ling, and Hao Su. Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search. *ACM Transactions on Graphics (TOG)*, 41(4):1–18, 2022.

Zhaoming Xie, Xingye Da, Buck Babich, Animesh Garg, and Michiel van de Panne. Glide: Generalizable quadrupedal locomotion in diverse environments with a centroidal model. In *International workshop on the algorithmic foundations of robotics*, pp. 523–539. Springer, 2022.

Yinzhen Xu, Weikang Wan, Jialiang Zhang, Haoran Liu, Zikang Shan, Hao Shen, Ruicheng Wang, Haoran Geng, Yijia Weng, Jiayi Chen, et al. Unidexgrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Wenhao Yan and Jie Chu. Foundationpose++: Simple tricks boost foundationpose performance in high-dynamic scenes. `https://github.com/teal024/FoundationPose-plus-plus`, 2025.

Max Yang, Chenghua Lu, Alex Church, Yijiong Lin, Christopher J. Ford, Haoran Li, Efi Psomopoulou, David A. W. Barton, and Nathan F. Lepora. Anyrotate: Gravity-invariant in-hand object rotation with sim-to-real touch. In *Conference on Robot Learning (CoRL)*, 2024.

Weirui Ye, Fangchen Liu, Zheng Ding, Yang Gao, Oleh Rybkin, and Pieter Abbeel. Video2policy: Scaling up manipulation tasks in simulation through internet videos. *arXiv preprint arXiv:2502.09886*, 2025.

Patrick Yin, Tyler Westenbroek, Zhengyu Zhang, Joshua Tran, Ignacio Dagnino, Eeshani Shilamkar, Numfor Mbiziwo-Tiapo, Simran Bagaria, Xinlei Liu, Galen Mullins, Andrey Kolobov, and Abhishek Gupta. Emergent dexterity via diverse resets and large-scale reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2026.

Haoqi Yuan, Ziye Huang, Ye Wang, Chuan Mao, Chaoyi Xu, and Zongqing Lu. Demograsp: Universal dexterous grasping from a single demonstration. In *International Conference on Learning Representations (ICLR)*, 2026.

Hui Zhang, Zijian Wu, Linyi Huang, Sammy Christen, and Jie Song. Robust dexterous grasping of general objects. In *Conference on Robot Learning (CoRL)*, 2025.

Huayi Zhou, Ruixiang Wang, Yunxin Tai, Yueci Deng, Guiliang Liu, and Kui Jia. You only teach once: Learn one-shot bimanual robotic manipulation from video demonstrations. In *Robotics: Science and Systems (RSS)*, 2025.

Xinghao Zhu, Zixi Liu, Shalin Jain, Chenran Li, Milad Noori, Michael Andres Lin, Huihua Zhao, John Welsh, Mrinal Verghese, Wei Liu, Tingwu Wang, Xingye Da, Zhengyi Luo, Vishal Kulkarni, Naema Bhatti, Yuke Zhu, Linxi Fan, Bowen Wen, Danfei Xu, Soha Pouya, and Yan Chang. Learning dexterous manipulation using contact wrench guidance from human demonstration. *arXiv preprint arXiv:2607.00033*, 2026.

Yifeng Zhu, Arisrei Lim, Peter Stone, and Yuke Zhu. Vision-based manipulation from single human video with open-world object graphs. *Autonomous Robots*, 2024.

René Zurbrügg, Andrei Cramariuc, and Marco Hutter. Graspqp: Differentiable optimization of force closure for diverse and robust dexterous grasping. In *Conference on Robot Learning (CoRL)*, 2025.

APPENDIX

A Additional Related Works **17**

B Task Abstraction as a Stage-wise Scene Graph **17**

B.1 Predicate Vocabulary 18

B.2 VLM Query 18

B.3 Asset Preparation and Functional Regions 23

B.4 Physical Properties and Functional Region Localization 23

C Scene-graph-grounded Reinforcement Learning **24**

C.1 Observations 24

C.2 Rewards 25

C.3 Reset Generation 26

C.3.1 Predicate Samplers and Placement Order 27

C.3.2 Grasp Synthesis 27

C.3.3 Transition Feasibility and Physical Validity 28

C.4 Hierarchical decomposition with a QP low level. 29

C.5 Simulation Training Details 30

D Real-World Perception **32**

E Additional Experimental Results **32**

E.1 Generalization to Unseen Initial Object Positions 32

E.2 Diverse Grasp Strategies Across Embodiments and Configurations 36

E.3 Simulation Rollouts Across Diverse Dexterous Embodiments 38

A Additional Related Works

Demonstration-seeded RL in simulation. A parallel line makes multi-stage exploration tractable by injecting demonstration-derived information into RL in simulation. Prior methods use this information as an action prior (Torne et al., 2024; Ankile et al., 2025), as object-pose progress targets (Dan et al., 2025), or to generate and refine task-specific reset and reward programs (Ye et al., 2025). In each case, the demonstration guides exploration by specifying how the policy should act, progress, or be rewarded. Instead, we use the demonstration to identify how object relations change at the predicate level encoded in the stage-wise scene graph over time. These changes define the task stages, from which we construct diverse, physically valid reset states across all intermediate stages, including synthesized multi-finger grasps. These stage-wise reset states allow episodes to start directly from intermediate stages of the task, reducing exploration burden. Furthermore, exposing the policy to multiple configurations results in a more generalizable policy. Thus, the video specifies the stage-wise relations the policy must achieve and their temporal order, rather than providing demonstrated actions, motion targets, or a reward program to follow.

Task-Aware Grasp Synthesis and Learned Grasp Acquisition. TaskDexGrasp conditions static dexterous hand-pose synthesis on task semantics, while Dexterous Functional Grasping combines semantic affordances with a learned grasp-acquisition controller (Chen et al., 2024; Agarwal et al., 2023). A broader line learns generalizable grasp proposal, acquisition, or maintenance from synthetic grasps and demonstrations (Xu et al., 2023; Wan et al., 2023; Yuan et al., 2026; Park et al., 2026; Zhang et al., 2025). These methods primarily target robust grasp acquisition across diverse objects, whereas our goal is to solve the full manipulation task. We use grasp synthesis to generate diverse reset states and train the policy via reinforcement learning to acquire the grasp and execute subsequent manipulation stages.

Sim-to-Real Dexterous Manipulation. Simulation-based reinforcement learning has been successfully applied to real-world dexterous in-hand orientation (Andrychowicz et al., 2020). Subsequent work has extended this setting toward faster control (Handa et al., 2023), more accessible hardware, and tactile feedback (Yang et al., 2024). Recent systems move toward longer-horizon interaction. SimToolReal (Kedia et al., 2026) trains a policy to move tools toward target poses and executes unseen tasks by tracking the full 6D tool trajectory extracted from a human video. Because this target pose sequence is assumed available during execution, and our task setting does not provide such trajectory information at execution time, the two settings are not directly comparable. Play2Perfect (Lum et al., 2026) and ADEPT (Lee et al., 2026b) instead focus on reusable dexterous pretraining: Play2Perfect learns from task-agnostic play before fine-tuning on downstream assembly tasks, while ADEPT pretrains generic object-reposing skills and adapts them to insertion and placement. These approaches complement to ours, as they study how pretraining supports downstream manipulation, whereas we focus on learning a new task from a single human video.

Force Closure and Grasp-Quality Objectives. Force closure is widely used to evaluate and synthesize stable multi-finger grasps. Recent methods formulate differentiable force-closure objectives to generate physically stable dexterous grasps across objects and hand embodiments (Liu et al., 2021; Wang et al., 2023b; Zurbrügg et al., 2025).

Force-closure measures have also been used as reinforcement learning rewards. CHORD (Zhu et al., 2026) introduces a force-closure reward for learning dexterous manipulation when demonstration-derived contact guidance is unreliable. We adopt this force-closure term by augmenting it with a weakest-direction grasp-quality measure to encourage stable grasps. While concurrent work similarly uses grasp quality as a dense reward to strengthen weak wrench directions during in-hand manipulation (Chen et al., 2026b), we augment the force-closure reward with a weakest-direction grasp-quality measure to acquire stable grasps as part of a multi-stage manipulation task.

B Task Abstraction as a Stage-wise Scene Graph

This section details how we obtain the stage-wise scene graph from a human video. Sec. B.1 lists the predicate vocabulary. Sec. B.2 describes how a single VLM query converts the video into the stage-wise scene graph, including the input prompt, an example output, and the query time and cost. Sec. B.3 describes how we reconstruct object assets from the demonstration, estimate their

physical properties, and localize their functional regions. The pretrained models used at each step are summarized in Table 2.

Table 2: Pretrained models used in task abstraction, asset preparation, and real-world perception.

Step	Model	Input → Output	Sec.
Scene-graph extraction	Gemini 3.1 Pro	video → entities, parts, stage sequence	B.2
Instance segmentation	SAM 3.1	first frame, `visual_phrase` → object mask	B.3
Mesh reconstruction	SAM 3D Objects	RGB, mask, metric point map → metric mesh	B.3
Object pose tracking	FoundationPose++	RGB-D stream, mesh → 6D pose	D
Physical parameters	GPT-5	category, size → mass, friction	B.4
Part localization	SAM 3	rendered views, part name → part region	B.4

Table 3: **Predicate vocabulary.** Geometric features, distance metrics, and constraints used to instantiate the predicates in our tasks. Paired pre-relation and terminal predicates share the same distance metric and differ only in their constraints, except for `pre_grasp`/`grasp`.

Predicate	**Reference** A	**Target** B	**Distance metric**	**Constraint**
`pre_on_top` `on_top`	top support disk $\mathcal{D}_A$	bottom point p_B	$d_{\text{top}} = d(p_B, \mathcal{D}_A)$	$\epsilon_{\text{top}} < d_{\text{top}} \le \varphi_{\text{top}}$ $d_{\text{top}} \le \epsilon_{\text{top}}$
`pre_inside` `inside`	interior box $\mathcal{B}_A$	root point p_B	$d_{\text{in}} = d(p_B, \mathcal{B}_A)$	$\epsilon_{\text{in}} < d_{\text{in}} \le \varphi_{\text{in}}$ $d_{\text{in}} \le \epsilon_{\text{in}}$
`pre_contact` `contact`	center p_A	working point p_B^{work}	$d_{\text{con}} = \lVert p_A - p_B^{\text{work}} \rVert_2$	$\epsilon_{\text{con}} < d_{\text{con}} \le \varphi_{\text{con}}$ $d_{\text{con}} \le \epsilon_{\text{con}}$
`pre_seat` `seat`	goal q_{goal}	joint position q_B	$d_{\text{seat}} = \lvert q_B - q_{\text{goal}} \rvert$	$\epsilon_{\text{seat}} < d_{\text{seat}} \le \varphi_{\text{seat}}$ $d_{\text{seat}} \le \epsilon_{\text{seat}}$
`pre_grasp` `grasp`	grasp-region p_A^{grasp} grasp set $\mathcal{G}_A$	fingertips P_B^{tip} hand configuration h_B	$d_{\text{pre-grasp}} = d_{\text{tip}}(P_B^{\text{tip}}, p_A^{\text{grasp}})$ –	$\ell_g \le d_{\text{pre-grasp}} \le u_g$ $h_B \in \mathcal{G}_A$

Notation. $d(p, S) = \min_{s \in S} \lVert p - s \rVert_2$ denotes the Euclidean distance from a point p to a set S. $\mathcal{D}_A$ and $\mathcal{B}_A$ denote the top support disk and interior box of reference entity A, respectively. For relation c, ϵ_c denotes its satisfaction tolerance, while φ_c denotes the outer sampling bound of the corresponding pre-relation. P_B^{tip} denotes the fingertip positions, and d_{tip} denotes the fingertip-to-grasp-region-center distance used for `pre_grasp`. ℓ_g and u_g denote its lower and upper sampling bounds. $\mathcal{G}_A$ denotes the verified grasp set for object A, and h_B denotes the hand configuration.

B.1 Predicate Vocabulary

Table 3 lists the predicates in our vocabulary, which cover common relations in tabletop manipulation. Each geometric predicate also has an approach form, denoted by the prefix `pre-`, which holds while the relation is being approached but not yet established.

B.2 VLM Query

Given the single human demonstration of each task (Fig. 8), recorded at 640×480, we query Gemini 3.1 Pro (`gemini-3.1-pro-preview`) (Gemini Team, Google, 2026) with the entire video at medium media resolution; videos under 20,MB are sent inline, and larger ones through the file-upload API. The prompt (Listing 1) is identical for all tasks: the only task-specific input is the video, and the vocabulary in the prompt is the one in Table 3. A single query takes 12–21s and costs less than $0.05, so extracting all five task graphs takes under 90s and $0.20 in total (Table 6).

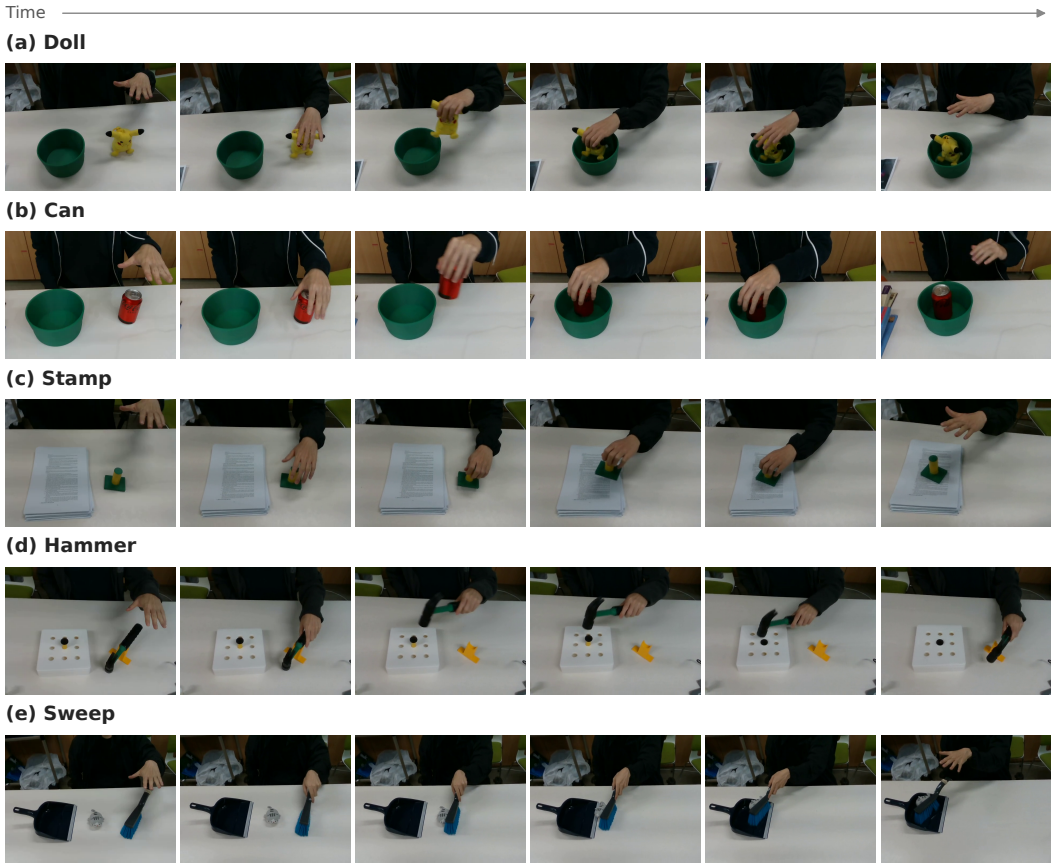


Figure 8: **Single human demonstrations for the five tasks.** Each row shows the human video used to specify one task, with frames ordered from left to right over time.

Table 4: **Task suite.** Object roles: *held* is grasped by the robot, *site* receives another object, *acted-on* is manipulated only through a tool, and *fixture* holds an object at the start. The predicate chain extracted by the VLM defines K stages, each with its own reset distribution. Objects marked with † use a CAD mesh instead of a reconstructed one (Appendix B.3).

Task	Description	Objects (role)	K	Predicate chain (stage 1 $\rightarrow$ K)	Type
Doll	Pick up a Pikachu plush toy and place it inside an orange pot.	pikachu (held), pot† (site)	5	$\varnothing$ $\rightarrow$ `pre-grasp`(hand, pikachu) $\rightarrow$ `grasp`(hand, pikachu) $\rightarrow$ `pre-inside`(pikachu, pot) $\rightarrow$ `success:inside`(pikachu, pot)	direct
Can	Pick up the coke can and place it in a bucket.	can (held), bucket† (site)	5	$\varnothing$ $\rightarrow$ `pre-grasp`(hand, can) $\rightarrow$ `grasp`(hand, can) $\rightarrow$ `pre-inside`(can, bucket) $\rightarrow$ `success:inside`(can, bucket)	direct
Stamp	Grasp the stamp and press it down onto the stack of papers.	stamp† (held), paper (site)	5	$\varnothing$ $\rightarrow$ `pre-grasp`(hand, stamp) $\rightarrow$ `grasp`(hand, stamp) $\rightarrow$ `pre-on_top`(stamp, paper) $\rightarrow$ `success:on_top`(stamp, paper)	direct

Task suite. (Continued)

Task	Description	Objects (role)	K	Predicate chain (stage $1 \rightarrow K$)	Type
Hammer	Use a hammer to strike a peg inserted into a block.	hammer† (held), peg† (acted-on), block† (site), holder† (fixture)	6	`fixed_to`(hammer, holder) → `pre-grasp`(hand, hammer) → `grasp`(hand, hammer) → `pre-contact`(hammer, peg) → `contact`(hammer, peg) → `success`:**`seated`**(peg, block)	tool-use
Sweep	Sweep the toy cat into the dustpan using the brush.	brush (held), toy (acted-on), dustpan (site)	7	∅ → `pre-grasp`(hand, brush) → `grasp`(hand, brush) → `pre-contact`(brush, toy) → `contact`(brush, toy) → `pre-inside`(toy, dustpan) → `success`:**`inside`**(toy, dustpan)	tool-use

The VLM returns a JSON object following a fixed response schema, whose fields and their use in our pipeline are summarized in Table 5. Listing 2 shows the output for the Sweep task, in which a brush is used to push a toy cat into a dustpan, and Table 4 summarizes the resulting objects, roles, and stage sequences for all five tasks. Table 4 lists the objects, their roles, and the stage sequence extracted by the VLM from each video.

Table 5: Structure of the VLM output and the use of each field in our pipeline. Indented entries are fields of the object above them, and `[]` marks a list with one entry per object, mount, or stage.

Field	Content	Used for
`task`		
`name`, `description`	Task name and one-line summary	Generated first to condition the subsequent fields; not used by the pipeline
`objects[]`		
`id`	Identifier of each task-relevant object	Nodes; key for all per-object data
`category`	Object class	Hint in the physical-parameter query
`visual_phrase`	Text description of the object's appearance	Object segmentation
`roles`	Role of the object (e.g., held, site)	Scene slots in reset generation and RL
`upright`	Whether the object is normally kept upright	Restricts sampled stable poses to the upright class
`grasping_part`, `working_part`	Names of functional parts	Text prompts for functional-region localization
`working_direction`	How a tool meets its target (down or horizontal)	Contact placement
`scene_graph`		
`initial_state[]`	Relations that already hold in the first frame	First stage graph; kept in the resets of later stages until the related object moves
`stages[]`	One predicate per stage, in temporal order	Stage sequence for backward reset generation; the last stage sets the near-goal target
`goal_condition`	Predicates that must all hold at success	RL success test (first predicate)

Listing 1: VLM prompt for stage-wise scene graph extraction. {`predicate_list`} is filled with the predicates of Table 3.

```
You are analyzing a human demonstration video for robot learning (real2sim).
The human hand in the video corresponds to a ROBOT END-EFFECTOR. Refer to it as "ee".
Do NOT list the hand as an object; it is the agent, not a manipulable object.

Extract, grounded ONLY in what is visually observable:

1. TASK: concise name and one-line description.

2. OBJECTS: the manipulable objects of interest (NOT the hand). For each: a short id,
   a category, a visual_phrase, and roles.
   category vs visual_phrase | these are DIFFERENT and both required:
   - "category": the clean object class used to search an asset library ("kettle", "mug",
     "knife"). A superordinate is fine here.
   - "visual_phrase": what you would type into an open-vocabulary SEGMENTATION model to
     select exactly this object in the video frame. 2-4 words, concrete and appearance-
     bearing: colour and/or material plus the specific noun ("red enamel kettle", "white
     ceramic mug", "wooden cutting board"). NEVER an abstract superordinate ("container",
```

```
    "object", "item", "tool") | those ground to nothing. If two objects in the frame share
    a noun, the phrase must be what tells them apart.
  Do NOT include the workspace surface itself (worktable, desk, floor, walls) | it
  is part of the robot cell, not an object. But DO include a fixed object that holds
  another object off that surface (a knife holder, a dish rack, a tray): it is not
  manipulated, yet the scene is wrong without it.
  Likewise include every object that another object is, or ends up, "seated" in
  (inserted, driven, or screwed into it: the bottle a cork is pressed into, the
  socket a plug is pushed into), even if the hand never touches it.
  roles | assign ALL that apply, judged from what visibly happens in the video:
  - "held":     the hand grasps and holds this object at some point
                (both directly-placed objects and tools count).
  - "site":     another object comes to rest ON or IN this object
                (plate under food, cutting board, container receiving something).
  - "acted_on": a HELD object acts on this object through contact
                (food being cut, dough being rolled, a window being wiped).
  - "fixture":  this object is already holding another object up when the demo
                begins, and the hand never touches it (a holder, a rack, a tray).
  "site" and "fixture" differ in WHEN: a site receives an object as a result
  of the manipulation, a fixture is holding one before it starts. The plate a
  cake is placed on is a site; the holder the knife rests in is a fixture.
  A pot that is carried and later receives food is ["held", "site"].
  Also determine per object:
  - "upright": true if the object is normally stored/used standing upright
    (cup, bottle, vase, jar → true; plate, cutting board, spatula, knife → false).
  - "grasping_part": for "held" objects, the part the hand grasps.
  - "working_direction": for "held" tool-use objects, how the working part moves
    against the target at the moment of contact | "down" if the tool is raised and
    brought down onto it (pestle, potato masher, cookie cutter), "horizontal" if the tool
    ↪  is pushed
    sideways across a surface (squeegee, spatula, rolling pin). This decides the
    contact pose the sampler builds; getting it wrong produces a tool that slides sideways
    when it should come down. null if not tool-use.
  - "working_part": for "held" objects that physically contact another
    object to do work (tool-use), the functional surface that does the work.
    null if the held object is placed directly (not tool-use).

  PART NAMING | these two strings are fed verbatim to an open-vocabulary
  segmenter as text prompts. Name the SMALLEST region that actually carries
  the part's role, and describe ONLY that region.

  Write "<attribute> <part> of the <object>" with EXACTLY ONE attribute, and
  that attribute must belong to the region you are naming | not to anything
  attached to it. If the handle is the part, describe the handle alone: its
  own colour, its own material. Say nothing about the head, the shaft, the
  body, or the object's overall look, even when those features are the most
  obvious thing in the demo. Every word describing a neighbouring region is
  an invitation for the mask to spread into that region.

  Keep the <object> slot bare | the plain object noun, never "the red mug"
  or "the wooden-handled knife".

  NEVER enumerate. If the region itself shows two colours, pick the one that
  covers most of it | do not write both. A colour the region shares with a
  neighbouring region is fine: name the region as it actually looks. Never
  borrow a colour from an adjacent part just to stand apart from it.
      good: "blue handle of the screwdriver"  (grip region only)
            "black handle of the spatula"     (same black as the blade | still
                                               right, that is what it looks like)
            "rubber blade of the squeegee"    (blade only, not the frame)
            "metal blade of the knife"
            "black grip of the pan"
      bad:  "handle"                          (no attribute | bleeds outward)
            "silver handle of the ladle"      (silver is the neck; the grip
                                               the hand holds is black | a colour
                                               borrowed from the neighbour)
            "black handle of the silver pan"  (attribute on the object |
                                               mask jumps to the body)
            "wooden handle of the metal knife"  (describes the blade too)
            "blue and red handle of the screwdriver" (enumeration | matches
                                                      the whole screwdriver)
            "blue rubber ergonomic handle of the screwdriver"  (piled attributes)
  Use only attributes actually visible in the demo; never invent a colour.
  If no attribute applies to the region on its own, write just
  "<part> of the <object>". null if not applicable.

3. SCENE_GRAPH: express it with PREDICATES ONLY (do not write free-form sentences),
  in three parts: initial_state, goal_condition, stages.

  initial_state: what is already mounted on what in the FIRST frame, before the
  hand moves. Most demos start with everything simply lying on the table | in
  that case return an empty list. Emit an entry only when an object begins the
```

```
    demo **held off the table by another object** in a definite resting pose: a
    ladle hanging on its rack, a part seated in a tray, a bit sitting in a holder.
    The only predicate here is "fixed_to": a is the object being held, b is the
    thing holding it. Nothing else | this list is not about what the hand does.
    - arg 'a' must be an object that LATER MOVES (it appears in the
      stages). An object that never moves needs no entry: it is simply part
      of the scene.
    - arg 'b' is the holder. Give it the "fixture" role in OBJECTS.
    - Do not describe an object resting directly on the table or the ground.
    - Do not describe a part that is already inserted, driven, or screwed into
      another object. That is a joint, not a mount, and it is not expressed here.

    goal_condition and stages | allowed predicates: {predicate_list}.
    - goal_condition: the set of predicates that must ALL hold at success (goal state).
    - stages: the relations that become established during the demo, one per entry,
      in the temporal order in which they are established. List only relations that
      become TRUE; do NOT add entries for approaching, reaching, or moving toward
      something | approach phases are derived automatically from these relations.
      Use "ee" as arg 'a' for grasp.
    Be precise about when each relation is actually established (e.g. distinguish
    real contact from the tool merely hovering near the target).
    AGENCY TRANSFER: when a source acts on a target through "grasp" or "contact", the
    target becomes arg 'a' of the subsequent stages | it is the thing that moves
    next, not the original actor. (ee grasps spatula → spatula becomes arg 'a'; spatula
    ↪  contacts
    pancake → pancake becomes arg 'a' for the move onto the plate.)
    Choose predicates from the list; never invent new ones.
    - "inside" vs "seat": use "seat" ONLY when the object ends up DRIVEN/PRESSED INTO
      the other one so the two become a single rigid assembly (a cork pressed into a
      bottle, a plug pushed into a socket, a drawer pushed shut). Use "inside" when the
      object merely ends up within an open container and could be lifted straight out
      (fruit placed in a basket, an item dropped in a bin).
    (grasp->grasp, contact->contact, place->on_top/inside/seat).

Do not hallucinate objects not present. Return only valid JSON matching the schema.
```

Listing 2: VLM output for the Sweep task


```
{
  "task": {
    "name": "sweep toy into dustpan",
    "description": "Use a brush to sweep a small plush toy into a dustpan."
  },
  "objects": [
    {
      "id": "brush",
      "category": "brush",
      "visual_phrase": "blue brush",
      "roles": [
        "held"
      ],
      "upright": false,
      "grasping_part": "black handle of the brush",
      "working_part": "blue bristles of the brush",
      "working_direction": "horizontal"
    },
    {
      "id": "dustpan",
      "category": "dustpan",
      "visual_phrase": "blue dustpan",
      "roles": [
        "site"
      ],
      "upright": false,
      "grasping_part": null,
      "working_part": null,
      "working_direction": null
    },
    {
      "id": "toy",
      "category": "plush toy",
      "visual_phrase": "grey cat plush",
      "roles": [
        "acted_on"
      ],
      "upright": true,
      "grasping_part": null,
      "working_part": null,
      "working_direction": null
    }
```

```
  ],
  "scene_graph": {
  "initial_state": [],
  "stages": [
    {"step": 1, "predicate": "grasp",   "args": {"a": "ee",     "b": "brush"}},
    {"step": 2, "predicate": "contact", "args": {"a": "brush", "b": "toy"}},
    {"step": 3, "predicate": "inside",  "args": {"a": "toy",   "b": "dustpan"}}
  ],
  "goal_condition": [
    {"predicate": "inside", "args": {"a": "toy", "b": "dustpan"}}
  ]
}
}
```

Table 6: VLM query cost per task.

Task	Video [s]	Latency [s]	Cost [$]
Doll	7.07	21.2	0.045
Can	5.30	11.5	0.024
Stamp	4.97	12.0	0.026
Hammer	4.00	18.7	0.039
Sweep	5.77	18.5	0.037
Total	27.11	81.9	0.171

B.3 Asset Preparation and Functional Regions

We reconstruct each task-relevant object from the first frame of the demonstration; the full video is used only by the VLM query. The object's visual phrase from the VLM output serves as the text prompt for SAM 3.1 (Carion et al., 2026), which returns the object's instance mask. Given the RGB image, the mask, and the metric point map from the depth image of the same frame, SAM 3D Objects (Chen et al., 2026a) reconstructs the object mesh. For objects marked in Table 4, we instead use an available CAD mesh.

Each mesh, reconstructed or CAD, is post-processed into a watertight simulation mesh and a set of convex pieces (Table 7). The convex pieces serve as collision geometry for grasp synthesis, and the watertight mesh serves as the collision mesh in Isaac Sim (Mittal et al., 2025).

Table 7: Mesh post-processing.

Step	Tool	Parameters
Convex decomposition	CoACD (Wei et al., 2022)	concavity threshold 0.05; resolution 2000; MCTS 20 nodes, 150 iterations, depth 3; $rv_k = 0.3$; manifold preprocessing only for non-manifold input (resolution 50); hull merging on; no hull-count limit; minimum part volume 10^{-10} (`-mv`); random seed from the system clock
Watertight remesh	CoACD manifold mode (OpenVDB (Museth et al., 2013))	applied to the normalised mesh; dual-marching-cubes level set 0.1
Simplification	ACVD (Valette & Chassery, 2004)	2000 vertices, gradation 1.5, manifold output

B.4 Physical Properties and Functional Region Localization

The VLM query provides only semantic attributes of each object (Table 5); physical parameters and functional regions are obtained as follows.

Physical parameters. Mass, mass range, and friction coefficient are estimated after scene-graph extraction by a separate text-only language-model query (`gpt-5` (Singh et al., 2025)), which receives the id, category, and measured longest axis of each object(Listing 3).

Here, {`obj_lines`} holds one line per object: its id, category, and measured longest axis in cm. During RL, the object mass is sampled from $[m - \Delta m,\ m + \Delta m]$ and the static friction from $[0.9\mu,\ 1.1\mu]$ with a floor of 0.05, and the dynamic friction is set to 0.9 times the static friction (Table 10). Physical verification (Sec. acc:physical_verification) use the worst case of these ranges, namely the maximum mass and the minimum friction.

Table 8: Localization of grasping and working regions.

Step	Setting
Views	34 views: elevations $-35^\circ, 10^\circ, 35^\circ, 60^\circ \times 8$ azimuths (45° steps), plus top (85°) and bottom (-85°); rendered twice (textured and uniform grey); 768×768 px, vertical field of view 45°, camera distance $2.6\times$ the half bounding-box diagonal
Detection	SAM 3; the part string is one text concept; instance score floor 0.15, mask threshold 0.5
Negative concepts	grasping only: *blade, cutting edge, spike, bristles*; their masks are dilated by 12 px and removed
Fusion	a vertex is visible in a view if its depth is within 2% of the bounding-box diagonal of the z-buffer; score $=$ positive votes / visible views, over views with at least one detection; vertex kept if seen in ≥ 2 views and score > 0.3
Transfer to mesh	each simulation-mesh face takes the maximum score of the nearest triangle of the reconstructed mesh; face kept if score ≥ 0.25; grasping region: largest connected component
Outputs	per-face mask and object-frame bounding box (centre, extents) of the grasping region; for grasp synthesis on the table, the mask is restricted to faces whose lowest vertex is ≥ 5 mm above the table
Working surfaces	working-region faces clustered by adjacency with normal deviation $\leq 5^\circ$; area-weighted plane fit per cluster; clusters with RMS residual $\leq 6 \times 10^{-5}$ m and area $\geq 90\%$ of the largest are kept (centre, normal, half-extents)

Listing 3: Prompt for physical-parameter estimation

```
This is a domain-randomization setup for a robot simulation. The objects
below were 3D-scanned from real instances, so the longest-axis size is already fixed by
measurement -- do NOT re-estimate the size.

For each object, answering as "an object of that category whose measured size is that
↪  value",
give:
1. mass_g: typical mass at that size (g, empty / dry weight).
2. mass_range_g: +/- variation of the mass (g).
3. friction_coeff: static friction coefficient against a flat surface
   (plastic ~0.3, wood ~0.4, rubber ~0.8, ceramic ~0.4, metal ~0.3, fabric ~0.5).
Objects:
{obj_lines}

Answer in JSON, keeping each id exactly as given.
```

Grasp and working regions. We localize each part name from the VLM output on the object mesh in three steps (Table 8). We first render the mesh from 34 viewpoints around the object and segment the part in each view with SAM 3, using the part name as a text prompt. We then project the 2D masks back onto the mesh and keep the vertices that are detected in most of the views in which they are visible. Finally, the grasping region is taken as the largest connected component, and the working region is fitted with planar surfaces that represent the tool's working faces. To keep tool heads out of the grasping region, masks of generic tool-head concepts (e.g., blades and bristles) are removed from it. The grasping region restricts grasp synthesis and provides the approach target of the reward, and the working surfaces are used by the contact sampler and the tool reward.

Support patches. Support patches are the upward-facing flat regions of a mesh on which another object can rest. Faces whose normal has a vertical component above 0.99 (within about 8° of vertical) are grouped by height with a 5 mm tolerance, and groups with less than 5% of the largest area are dropped. Each patch stores its height, centre, and radius.

C Scene-graph-grounded Reinforcement Learning

In this section, we make an detail explanation for Sec. 3.2.

C.1 Observations

Scene-graph-grounded observation construction. The scene graph identifies the task-relevant entities, their roles, relations, and functional regions. We map the demonstrated hand to the robot end-effector and assign the remaining objects according to their roles in the task. For direct manipulation, the observation contains the end-effector, the manipulated object, and the receptive object. For tool-use manipulation, the held object is instead the tool, and an additional acted-on (interactive) object is introduced (Fig. 9).

Object and functional-part nodes provide their local geometry, such as object size, grasp regions, and working surfaces. Functional parts are associated with the hand or another object according to their role in the interaction. We represent a grasp region by its center and bounding-box extents, and a working surface by its center, unit normal vector of its surface, and the extent of the surface.

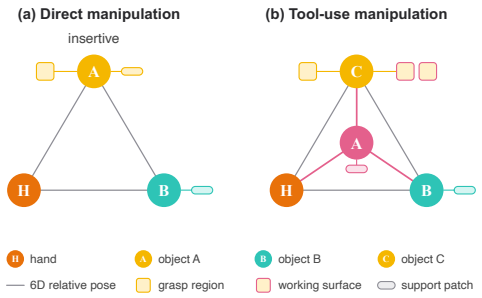


Figure 9: **Scene-graph-grounded observation.** The policy observation is constructed from scene-graph nodes and their relative 6D-pose edges for (a) direct and (b) tool-use manipulation.

Each scene-graph edge contributes a relative 6D pose between its connected entities. Importantly, this relative pose is defined according to the predicate encoded by the edge: the predicate determines which geometric features of the two entities are used, following the same feature definitions used for the predicate distance functions. Thus, an edge may relate object centers, functional regions, or other predicate-specific features rather than simply the two object centers. All node, part, and edge features are concatenated to form the policy observation.

C.2 Rewards

The `grasp` predicate is evaluated in 6D wrench space rather than by a Cartesian distance. Following CHORD (Zhu et al., 2026), we approximate the friction cone at each contact with a set of edge forces, each inducing a 6D primitive wrench consisting of force and torque. We then evaluate how well these primitive wrenches support a set of sampled directions in wrench space. Let $\sigma_{j,t}$ denote the support along sampled direction j at time t. CHORD's force-closure reward measures the fraction of directions whose support exceeds a threshold:

$$r_{\text{fc}}(s_t, a_t) = \frac{1}{B} \sum_{j=1}^{B} \mathbb{I}[\sigma_j > \epsilon].$$

where B is the number of sampled wrench directions and ϵ is the minimum support threshold.

In our setting, we use r_{fc} to guide grasp acquisition without a retargeted hand trajectory or virtual-object controller (Mandi et al., 2026). Directional coverage alone, however, can increase for contacts that do not form a stable grasp; for example, the policy may push the brush against the toy cat instead of first grasping the brush. We therefore augment r_{fc} with the Ferrari–Canny grasp quality (Ferrari & Canny, 1992), which measures support along the weakest wrench direction:

$$Q_t = \min_j \sigma_{j\,t}. \tag{1}$$

The resulting grasp reward is

$$r_{\text{grasp}}(o_t, a_t) = r_{\text{fc}}(o_t, a_t) + \mathbb{I}[N_{\text{finger},t} \geq 2] \max(Q_t, 0), \tag{2}$$

where $N_{\text{finger},t}$ is the number of distinct fingers in contact with the object. The additional quality term favors stable multi-finger grasps by encouraging support even along the weakest wrench direction.

Progress rewards from spatial predicates. For predicates whose relations can be expressed as a Cartesian distance, we define dense progress rewards from reductions in that distance. Following Kedia et al. (2026), we measure progress relative to the smallest distance reached so far in the episode. For `pre-grasp`(A, B), let $d_{AB,t}$ denote the current distance and $d^*_{AB,t}$ the smallest distance reached up to time t. We define

$$r_{\text{pre-grasp},t}(A, B) = \max\left(d^*_{AB,t-1} - d_{AB,t}, 0\right),$$

so that reward is produced only when the policy makes new progress toward the relation. Using improvement over the best distance reached so far avoids rewarding stationary behavior: once the distance stops decreasing, the progress reward becomes zero. We use the same progress construction for the other distance-based predicates. For `contact` predicate, we additionally include an alignment term between the tool's working region and the desired direction of interaction.

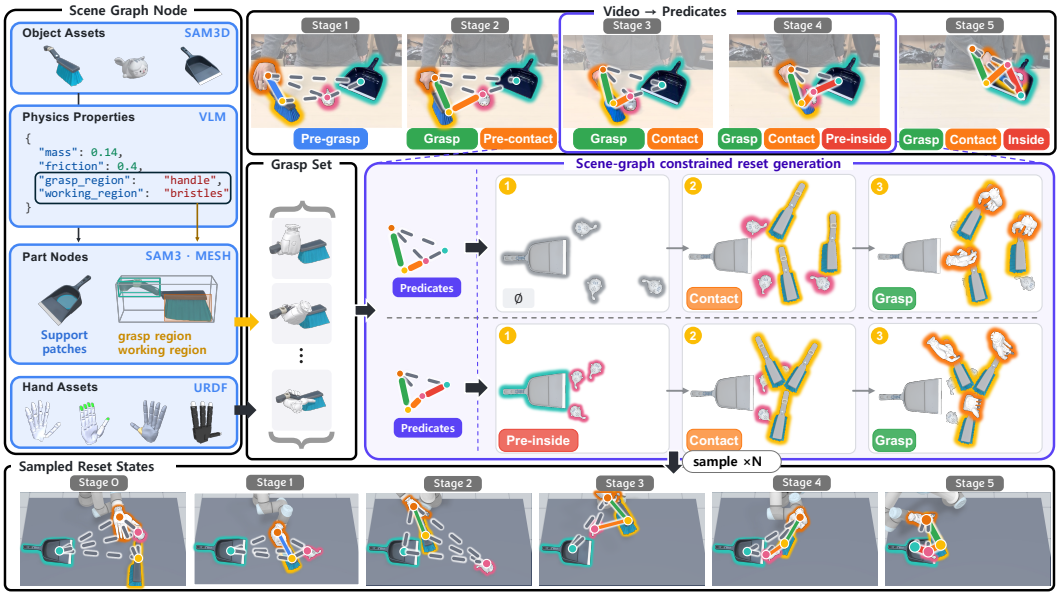


Figure 10: **Scene-graph-constrained reset generation.** Given the stage-wise scene graph and the assets, we instantiate each task stage as diverse reset states. The object and hand assets provide geometry and physical properties, while functional part nodes identify regions such as grasp and working regions. Together, these are used to construct a grasp set, restricted to the specified grasp region when available. For each stage, predicates in the scene-graph act as generative constraints: geometric predicates determine where related entities can be sampled, while the `grasp` predicate samples a valid hand–object configuration from the grasp set. Repeating this process produces diverse simulation reset states that satisfy the relations specified by each stage graph.

Task reward composition. We next describe how the reward terms are combined during training. We use a grasp indicator $\mathbb{I}_{\text{hold},t}$ to determine whether the object is currently lifted by a valid grasp. Using the same grasp-quality criterion as above, we define

$$\mathbb{I}_{\text{hold},t} = \mathbb{I}[N_{\text{finger},t} \geq 2]\,\mathbb{I}[Q_t > 0]\,\mathbb{I}[z_t - z_0 > h_{\text{lift}}]\,, \qquad h_{\text{lift}} = 5\,\text{cm}, \tag{3}$$

Thus, $\mathbb{I}_{\text{hold},t}$ is active only while at least two distinct fingers are in contact, the grasp provides positive support along its weakest wrench direction and the target is lifted above h_{lift}.

Let g denote the first stage whose scene graph contains a `grasp` predicate, and let $r_{i,t}$ denote the reward driven by the predicates associated with stage $\mathcal{G}_i$. At the grasp stage, we additionally use a lift reward before the hold condition is satisfied:

$$r_{g,t} = r_{\text{grasp},t} + \big(1 - \mathbb{I}_{\text{hold},t}\big) r_{\text{lift},t}. \tag{4}$$

The complete task reward is then

$$r_t = \sum_{i \leq g} r_{i,t} + \mathbb{I}_{\text{hold},t} \sum_{i > g} r_{i,t} + r_{\text{success},t} + r_{\text{reg},t}, \tag{5}$$

where rewards through the grasp stage are always available, while rewards associated with subsequent manipulation stages are activated only when the object satisfies the hold condition in Eq. 3. Here, $r_{\text{success},t}$ is the terminal success bonus, and $r_{\text{reg},t}$ penalizes large actions and joint velocities.

C.3 Reset Generation

This section details how we construct the stage-wise initial-state distribution ρ_0 of Sec. 3.2. Fig. 10 summarizes the process. Each stage graph $\mathcal{G}_k$ specifies which relations must hold at that point of the task but leaves object poses and grasps free, so many geometrically distinct states satisfy it. We generate such states in two steps. First, we sample states that satisfy the predicates of the stage graph,

using each predicate as a rule for placing one node relative to another (Sec. C.3.1); the `grasp` predicate is instantiated from a set of synthesized dexterous grasps (Sec. C.3.2). Second, we keep only the states that remain usable for the rest of the task and are physically valid in simulation (Sec. C.3.3). We retain 10,000 verified states per stage, which form its reset distribution. The resulting states are therefore not random configurations of the scene, but diverse instantiations of the same task structure.

C.3.1 Predicate Samplers and Placement Order

Each predicate in Table 3 is defined on geometric features of its two nodes, and the same features determine where one node can be placed relative to the other. For example, `on_top` restricts the bottom point of the source object to the support patch of the target, and `inside` restricts the root point of the source object to the interior box of the target. We use these conditions as placement rules: to satisfy a predicate, we sample the pose of its source node within the region that its target node and the predicate define.

To construct a full stage, we resolve its predicates in dependency order along the edges of $\mathcal{G}_k$. The anchor, a node that is the target but not the source of any predicate, is placed first. We then follow the edges from the anchor and place each source node relative to its already placed target. Finally, unconstrained nodes, which appear in no predicate of the stage, are sampled in the remaining free space. Fig. 10 illustrates this process for two stages of the Sweep task. The stage in the top row contains `contact`(brush, toy cat) and `grasp`(hand, brush). The toy cat is the anchor, so it is placed first; the brush is then placed relative to it so that its bristles touch the toy cat, and the hand is placed on the brush handle at a grasp from $\mathcal{H}_{\text{grasp}}$ (Sec. C.3.2). The dustpan appears in no predicate of this stage and is sampled last in the remaining free space. The stage in the bottom row additionally contains `pre-inside`(toy cat, dustpan), so the dustpan becomes the anchor: it is placed first, the toy cat is placed relative to it without being inside it, and the brush and the hand follow as above. For the final stage, we first sample a state that satisfies the goal predicate and then perturb it into a near-goal state, following OmniReset (Yin et al., 2026), since an already successful reset provides less useful learning signal.

C.3.2 Grasp Synthesis

A valid grasp is critical for stage-wise resets: every stage that contains `grasp` presupposes a securely held object, so an invalid grasp invalidates all of them. We synthesize grasp candidates with Dexonomy (Chen et al., 2025a), which takes a robot hand, an object mesh, and a grasp type from a 33-type taxonomy, and returns diverse grasps of that type. When the object has a grasp region (Appendix B.4), we restrict the synthesis to that region, so that, for example, the brush is grasped at its handle rather than its bristles. For each grasp, we also store the finger command q_{cmd} that closes the hand on the object and restore it at initialization together with the hand configuration. The corresponding `pre-grasp` state is obtained by linearly interpolating the finger joints toward the open configuration, to 30–90% of the grasp configuration, and retracting the hand by 1–5 cm along its approach axis.

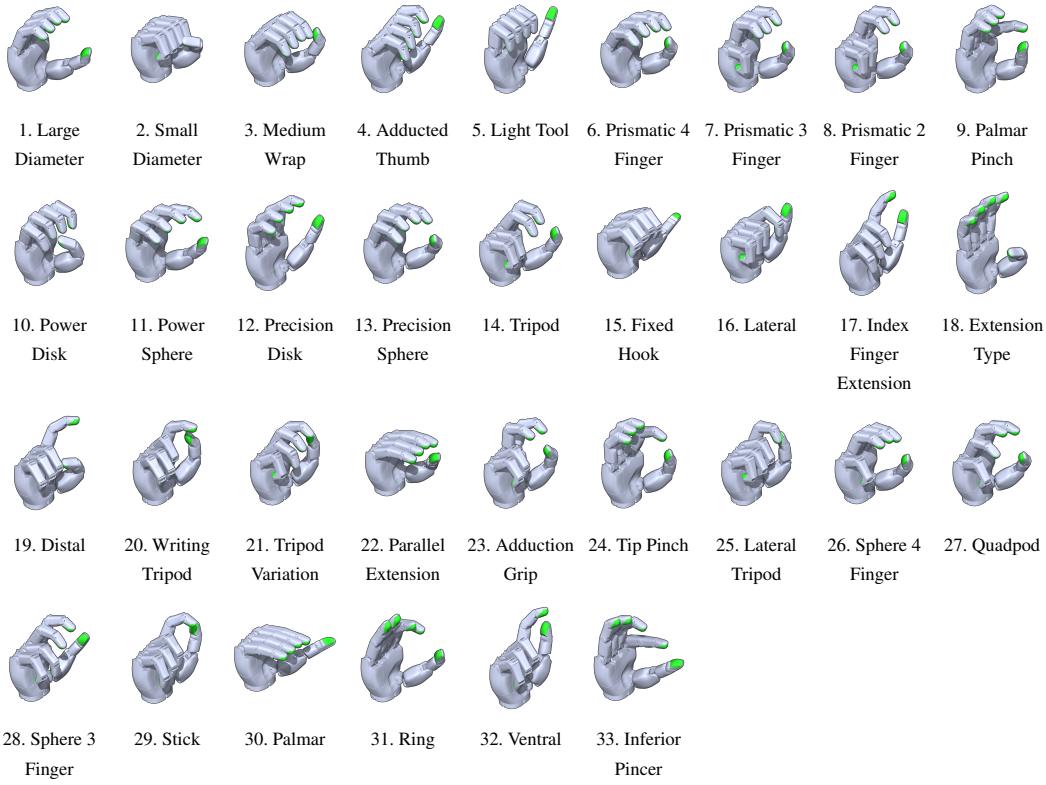


Figure 11: Grasp configurations of the Sharpa Wave hand.

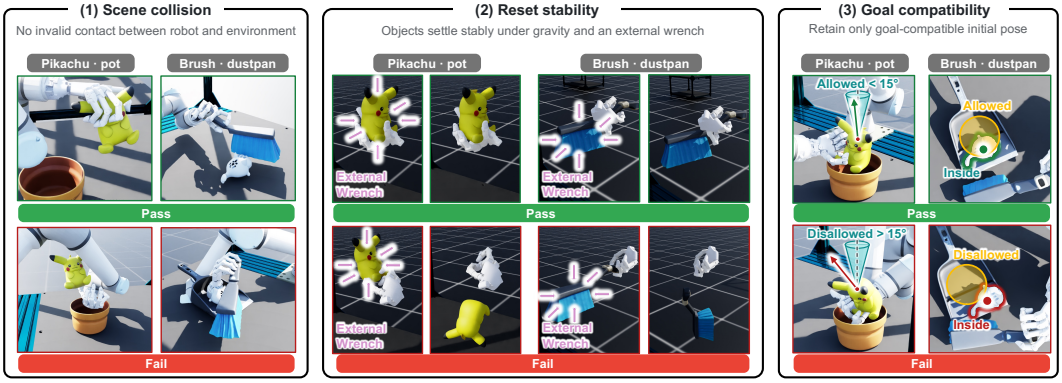


Figure 12: **Physical verification of reset candidates in the Sweep task.** (1) and (2) (top: pass, bottom: fail), and (3) is enforced over grasp types. (1) *Scene collision*: the robot makes no invalid contact with the table or other objects; the failing candidate collides with the dustpan. (2) *Reset stability*: objects settle under gravity and the grasped object stays in the hand under external wrenches; the failing grasp drops the brush. (3) *Goal compatibility*: the goal relation remains attainable with the candidate grasp; the failing candidate cannot bring the toy cat inside the allowed region of the dustpan.

C.3.3 Transition Feasibility and Physical Validity

Satisfying the predicates of one stage does not guarantee that a state remains usable later in the task. We therefore check compatibility at the grasp and stage levels, and verify every state physically.

Grasp level. A grasp that is stable in isolation may still place the hand in a configuration from which the rest of the task cannot be completed. Some grasps are valid when the object is picked up but fail in later stages. For example, when an object must be placed `inside` a narrow container, a grasp that wraps the whole object holds it securely at pickup, but brings the fingers into collision with the container at the goal. Because every stage after the grasp inherits the same grasp, such a grasp makes all of these stages unusable. We therefore select grasps by their compatibility with the whole task rather than by stability alone. For each grasp type, we instantiate its candidates in every stage that contains `grasp` and apply the physical checks below. We rank the grasp types by the fraction of candidates that pass across all such stages, and the candidates of the top-n types ($n = 4$) form the grasp set $\mathcal{H}_{\text{grasp}}$. Grasp types that hold the object but cannot attain the goal relation, such as the one in Fig. 12 (3), receive a low pass rate and are excluded. The resulting grasp set is therefore conditioned on the entire task, not only on how well each grasp holds the object.

Stage level. Sampling each stage independently can produce states that satisfy their own stage graph yet cannot lead to the next stage. For example, a `pre-grasp` state might place a pan with its handle pointing away from the robot, where no verified grasp of the next stage is reachable. Such states are valid locally but ill-posed for the task as a whole. We prevent them by generating the stages backward, starting from the goal stage. After the states of stage k are verified, we carry the poses of their anchor objects and their grasps into stage $k-1$, and resample only the remaining entities under the predicates of stage $k-1$. Each state of stage $k-1$ is thus constructed around a physically verified state of stage k rather than in isolation, so the policy can always progress from one stage to the next.

Physical validity. Every candidate state is instantiated in simulation with the full robot and must pass two checks (Fig. 12). (1) *Scene collision*: the robot makes no invalid contact with the table or other objects. (2) *Reset stability*: at the end of a 6 s episode, all objects have been at rest for the last 2 s, and in grasped states the object is still held by at least two hand links. Goal compatibility is enforced at the grasp level rather than per state: grasp types whose candidates cannot attain the goal relation fail these checks in the goal stage and are removed by the ranking above (Fig. 12 (3)). All checks use the worst-case physical parameters of the domain randomization, the maximum mass and the minimum friction (Appendix B.4), so that verified states remain valid across the randomized range. Grasp robustness is verified once per grasp before reset generation: with only the hand and the object, the grasp must hold the object under gravity in six directions and under an external force of $\min(0.5\, m_{\max} g,\ 0.5\,\text{N})$ with a 0.02 m lever along each of the six axes. The pass rate of each grasp type across all grasped stages provides the ranking at the grasp level above, and only states that pass all checks are retained and propagated backward to the previous stage.

C.4 Hierarchical decomposition with a QP low level.

Zero-shot sim-to-real deployment can expose the robot to hardware-specific constraints that are difficult to anticipate during simulation training. For example, excessive joint velocities may lead to unsafe motions, while contacts that are benign in simulation can damage the physical system. In our setup, contact between the Wuji 1 hand coupler and the table can disengage the coupler from the robot.

To accommodate deployment-time safety constraints without retraining the policy, we separate learned task-space control from constrained joint-space execution. Following prior work (Xie et al., 2022; Gangapurwala et al., 2022; Bang et al., 2024), we let the RL policy act in task space and delegate kinematic resolution to a model-based low-level controller. Specifically, we adopt the QP velocity-IK formulation of Lee et al. (2026a), which maps the policy's palm and fingertip velocity commands to joint velocities subject to kinematic and safety constraints.

Controller formulation. We write superscripts for the frame in which a quantity is expressed and subscripts for the associated body or point. $\mathcal{B}$ and $\mathcal{P}$ denote the robot base and palm frames, and $\mathcal{F}$ denotes the set of fingertips. The policy outputs

$$a = \left(\xi_P^{\mathcal{B}}, \{v_i^{\mathcal{P}}\}_{i\in\mathcal{F}}\right), \tag{6}$$

where $\xi_P^{\mathcal{B}} \in \mathbb{R}^6$ is the desired palm twist expressed in the robot base frame and $v_i^{\mathcal{P}} \in \mathbb{R}^3$ is the desired linear velocity of fingertip i relative to the palm. The resulting action space has dimension $6 + 3|\mathcal{F}|$.

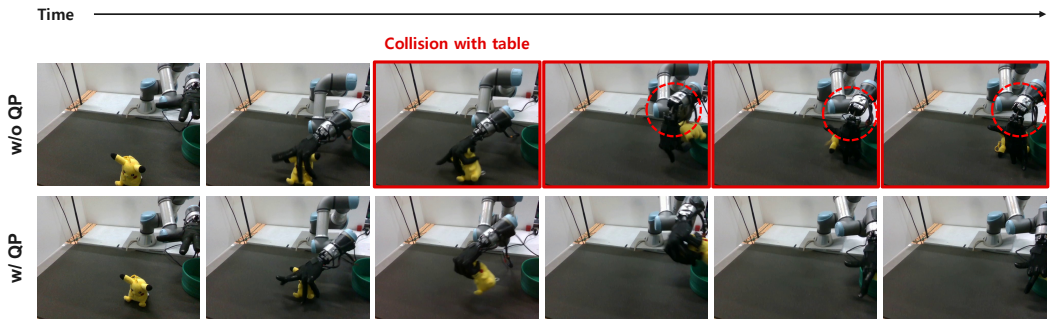


Figure 13: **QP constraints address a hardware-specific sim-to-real gap.** Hand–table contact is benign in simulation because coupler disengagement is not modeled, but the same contact can disengage the Wuji 1 hand coupler on the physical robot. Without the QP low-level controller (top), the learned policy encounters this failure mode. With the QP controller (bottom), a hand–table clearance constraint prevents the collision without retraining the task-space policy.

The low-level QP computes the joint velocity that best realizes these task-space commands:

$$\dot{q}_{\text{des}} = \arg\min_{\dot{q}} \ \left\|\xi_P^{\mathcal{B}} - J_P^{\mathcal{B}}(q)\dot{q}\right\|_2^2 + \sum_{i\in\mathcal{F}} \left\|v_i^{\mathcal{P}} - J_i^{\mathcal{P}}(q)\dot{q}\right\|_2^2 + \lambda\|\dot{q}\|_2^2 \tag{7}$$
$$\text{s.t.} \quad \varphi_c(q) + \nabla\varphi_c(q)^\top \dot{q}\Delta t \geq \varepsilon_{\text{col}}, \qquad c \in \mathcal{C},$$
$$q_{\min} \leq q + \dot{q}\Delta t \leq q_{\max}, \qquad \dot{q}_{\min} \leq \dot{q} \leq \dot{q}_{\max}.$$

Here, $J_P^{\mathcal{B}}$ maps joint velocities to the palm twist, while $J_i^{\mathcal{P}}$ maps them to the palm-relative velocity of fingertip i. Because the fingertip velocities are defined relative to the palm, the arm columns of $J_i^{\mathcal{P}}$ vanish, so the corresponding objective terms control only the finger joints. The constraints enforce collision clearance, joint-position limits, and joint-velocity limits over the control interval Δt. Here, φ_c denotes the signed distance for a monitored collision pair $c \in \mathcal{C}$ and ε_{col} is the required clearance.

Deployment-time constraint adaptation. The separation between task-space policy outputs and constrained joint-space execution allows us to modify the feasible joint motion at deployment without changing the learned policy. In particular, we can tighten $\dot{q}_{\min}$ and $\dot{q}_{\max}$ to reduce execution speed on the physical robot, while preserving the task-space behavior learned in simulation. We can also introduce additional collision constraints that were not required during simulation training.

For our real-world setup, we add a hand–table clearance constraint

$$\varphi_{\text{table}}(q) + \nabla\varphi_{\text{table}}(q)^\top \dot{q}\Delta t \geq \varepsilon_{\text{table}}, \tag{8}$$

where φ_{table} measures the minimum signed distance between the hand and the table and $\varepsilon_{\text{table}}$ specifies the required clearance. This constraint modifies only the QP at deployment; the RL policy remains unchanged.

Figure 13 illustrates why this separation is useful in our physical system. Without the hand–table clearance constraint, the learned policy can drive the Wuji 1 hand coupler into the table, which we observed to disengage the coupler during real-world execution. Adding the clearance constraint prevents this failure mode without additional policy training. Thus, the QP provides a mechanism for incorporating deployment-specific hardware and safety requirements while preserving the task-space policy trained in simulation.

C.5 Simulation Training Details

Simulation setup. We train all policies in Isaac Sim with Isaac Lab using the GPU PhysX pipeline. The simulator runs at 120 Hz, and the policy acts every 12 simulation steps, resulting in a control frequency of 10 Hz. Each episode lasts 16 s, corresponding to 160 policy steps. At every policy step, we solve the QP in Sec. C.4 in batch across all parallel environments to obtain the desired joint velocities. The four evaluated embodiments have 22–28 actuated joints and four or five fingertips. Accordingly, the task-space action dimension is 18 for four-finger hands and 21 for five-finger hands.

Table 9: **Policy optimization configuration.** We use the same configuration for all tasks and embodiments. The learning rate follows an adaptive KL schedule initialized at 10^{-4} and clipped to $[10^{-5}, 10^{-2}]$.

PPO		**Network**	
Parallel environments	49,152	Actor recurrence	LSTM, 1 layer, width 1024
Rollout horizon	32 steps (3.2 s)	Actor recurrent norm	Layer norm
Learning epochs	5	Actor MLP	$[1024, 1024, 512, 512]$
Mini-batches	4	Critic	MLP $[1024, 1024, 512, 512]$
Clip range	0.2	Activation	ELU
Discount γ	0.99	Action distribution	Gaussian, learned $\log \sigma$
GAE λ	0.95	Initial σ	1.0
Entropy coefficient	0.006	Observation history	1
Value-loss coefficient	1.0 (clipped)		
Optimizer	Adam		
Initial learning rate	1×10^{-4}		
Schedule	Adaptive, $\mathrm{KL}^* = 0.01$		
Gradient-norm clip	1.0		
Reset states per stage	10,000		

Table 10: **Domain randomization.** Multipliers are relative to the original simulation values. Ranges are sampled uniformly unless otherwise stated.

Property	**How it is varied**
Physical properties	
Object mass	Between the minimum and maximum mass estimated for each object
Object static friction	Up to 10% above or below the estimated friction
Object dynamic friction	Up to 10% above or below the estimated friction
Table friction	Static: 0.3–0.6; dynamic: 0.2–0.5
Table mass	0.5–1.5 times the original mass
Hand stiffness and damping	0.8–1.25 times the original values
Errors in actor observations	
object position	Gaussian error: 1 cm standard deviation per axis; capped at 3 cm per axis
static object position	Gaussian error: 0.5 cm standard deviation per axis; capped at 1.5 cm per axis
Object orientation	Rotate about a uniformly sampled direction by an angle between -5° and $+5^\circ$
Perceived object size	Up to 10% larger or smaller, equally in all directions
Delays	
Object-pose observation	0–150 ms
Robot-state observation	0–50 ms
Action execution	0–50 ms

Policy optimization. We optimize the policy with PPO (Schulman et al., 2017) under an asymmetric actor–critic. The actor receives only observations available at deployment and uses a single-layer LSTM with layer normalization followed by a four-layer MLP. The critic is memoryless and receives privileged simulation state. We use the same optimization and network configuration across all tasks and embodiments (Table 9).

Domain randomization. During simulation training, we vary physical properties and deliberately add noise to the actor's observations to improve robustness to real-world differences. Also we introduce observation and action delays to deal with sim2real gap. Table 10 summarizes these variations.

D REAL-WORLD PERCEPTION

The policy observes object poses rather than images (Sec. 3.2), so at deployment we estimate the 6D pose of each task-relevant object in real time. Two RGB-D cameras (Intel RealSense D435i) observe the workspace from complementary viewpoints, and FoundationPose++ (Yan & Chu, 2025) tracks each object independently in each camera, using the same mesh as in simulation. Because the hand frequently occludes the object during manipulation, a single view is often unreliable, and we fuse the two estimates as follows.

Multi-view pose fusion. We fuse the FoundationPose++ estimates T_1, T_2 from the two cameras according to how well each explains its own depth image. For camera c, we render the object mesh at T_c and compute the fraction of its projected silhouette Ω_c, including the part outside the image, whose sensor depth is valid and within τ of the rendered depth:

$$w_c = \frac{1}{|\Omega_c|} \sum_{u \in \Omega_c} \mathbb{I}\big[\, |D_c(u) - \hat{D}_c(u)| < \tau \,\big]. \tag{9}$$

Occluded or truncated views thus receive low weights. Because the attainable value differs between objects and cameras even without occlusion (a white sheet, for instance, returns no depth over much of its surface), each weight is normalized by its unoccluded baseline w_c^0, the median over the first 15 frames: $\bar{w}_c = \min\big(w_c / \max(w_c^0, 0.1),\, 1\big)$. The fused pose interpolates T_1 and T_2 with $\alpha = \bar{w}_2^k / (\bar{w}_1^k + \bar{w}_2^k)$, linearly in translation and by slerp in rotation. A camera is excluded when $\bar{w}_c < w_{\min}$ and readmitted only once $\bar{w}_c > w_{\mathrm{re}}$, and a pose that deviates from the higher-weighted one by more than 3 cm or 20° is left out of the blend, so that a lost track or a symmetric flip is never averaged. If both cameras are excluded, the pose is extrapolated with a constant-velocity model for up to 0.5 s and then held. The fused pose also serves as the next tracking prior of both cameras, which re-initializes a camera after an occlusion. We use $\tau = 1.5\,\mathrm{cm}$, $k = 3$, $w_{\min} = 0.25$, and $w_{\mathrm{re}} = 0.35$.

E ADDITIONAL EXPERIMENTAL RESULTS

E.1 GENERALIZATION TO UNSEEN INITIAL OBJECT POSITIONS

To further analyze spatial generalization beyond the human video, we fix the goal configuration and vary the initial object position across the feasible workspace. Figures 14–18 visualize initial-position generalization across the five evaluation tasks.

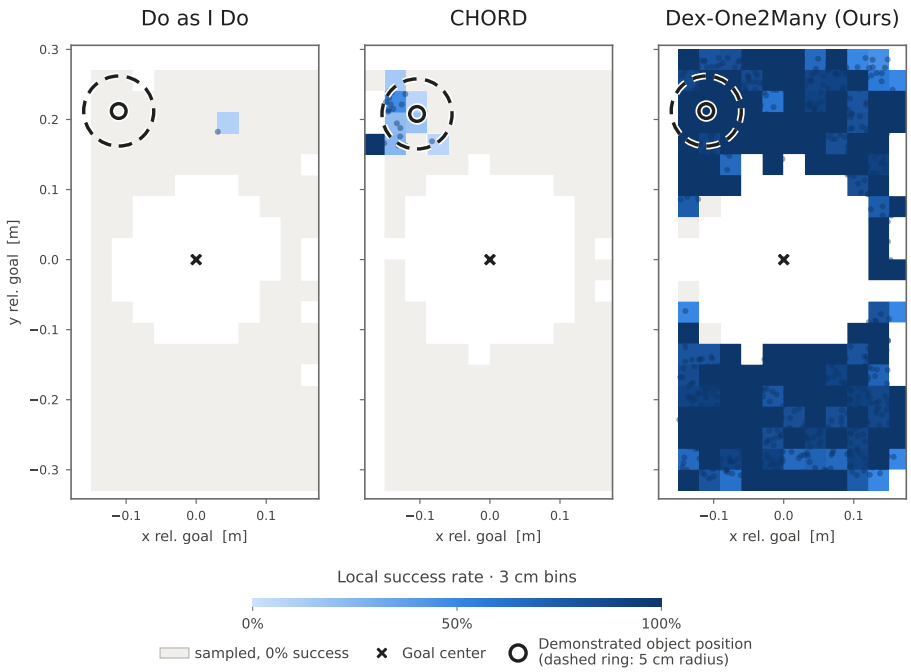


Figure 14: **Generalization to unseen initial object positions on the Doll task.** With the goal fixed, Dex-One2Many succeeds over a substantially broader range of initial positions than the baselines.

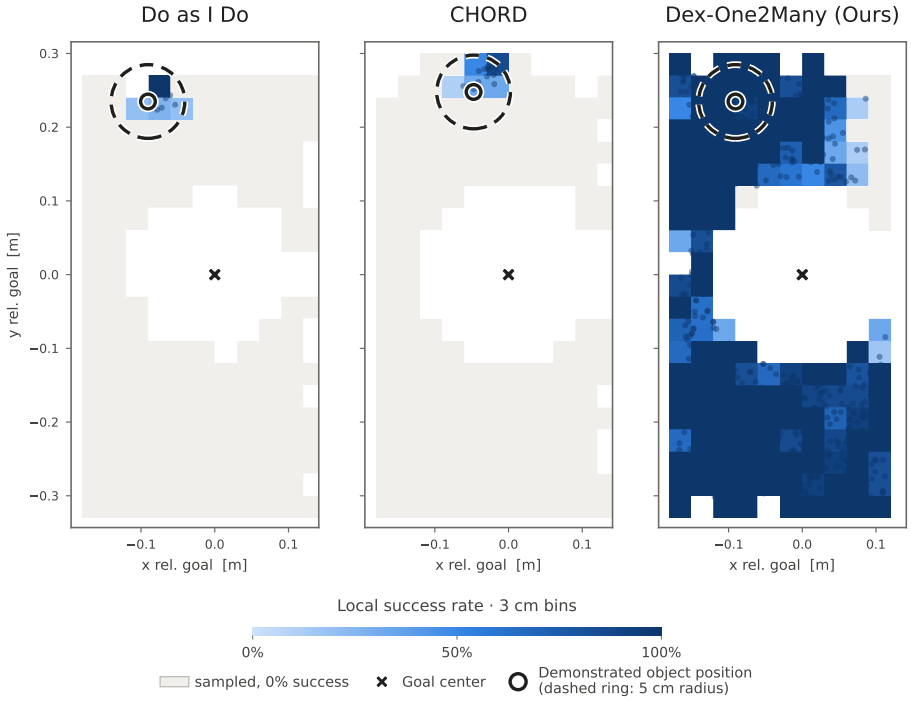


Figure 15: **Generalization to unseen initial object positions on the Can task.** With the goal fixed, Dex-One2Many succeeds over a substantially broader range of initial positions than the baselines.

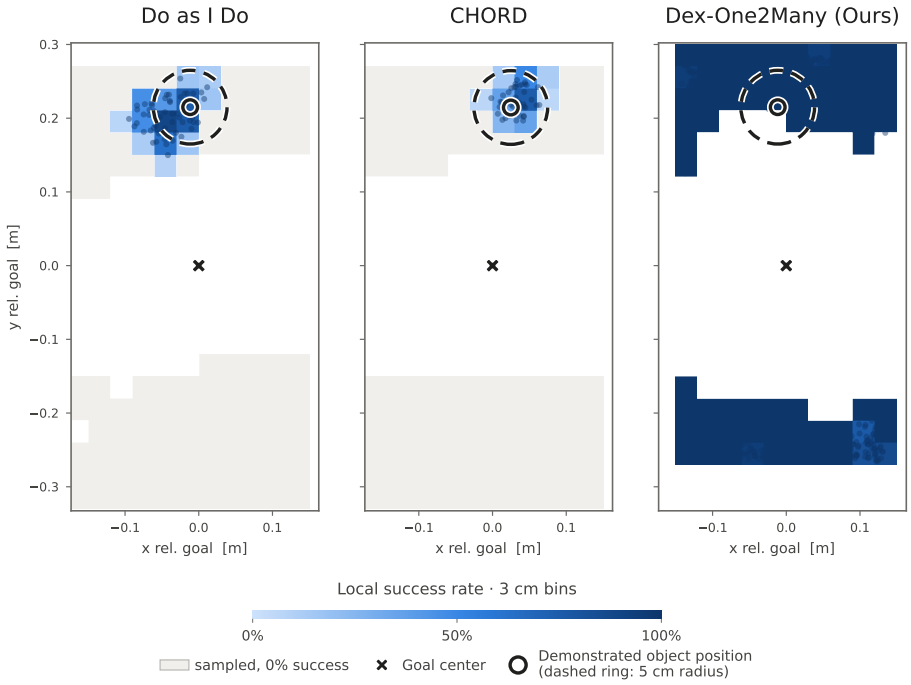


Figure 16: **Generalization to unseen initial object positions on the Stamp task.** With the goal fixed, Dex-One2Many succeeds over a substantially broader range of initial positions than the baselines.

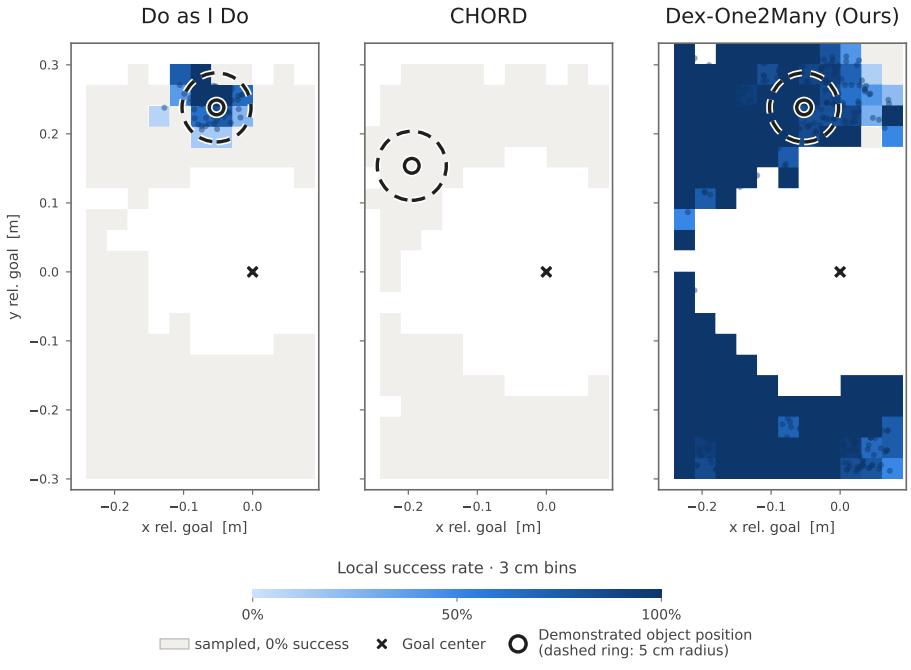


Figure 17: **Generalization to unseen initial object positions on the Hammer task.** With the goal fixed, Dex-One2Many succeeds over a substantially broader range of initial positions than the baselines.

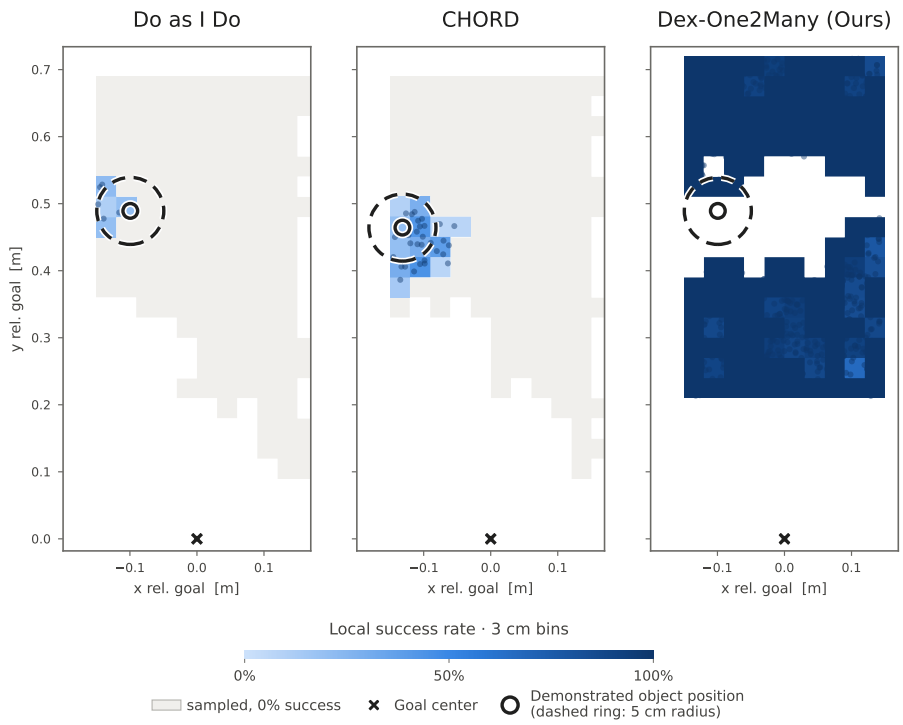


Figure 18: **Generalization to unseen initial object positions on the Sweep task.** With the goal fixed, Dex-One2Many succeeds over a substantially broader range of initial positions than the baselines.

E.2 Diverse Grasp Strategies Across Embodiments and Configurations

The task specification extracted from the human video does not prescribe a specific grasp configuration. Instead, our reset generation pipeline exposes the policy to diverse multi-finger grasps drawn from the 33-type grasp taxonomy, allowing the grasp strategy to adapt to the robot embodiment and the current object configuration.

Figures 19–21 show representative grasp strategies learned for the Doll, Can, and Stamp tasks. Different dexterous embodiments realize different grasp types for the same task, reflecting their distinct hand morphologies and kinematics. Moreover, even within the same embodiment, the learned grasp can change as the object configuration varies. These qualitative results illustrate that Dex-One2Many does not constrain the robot to a fixed grasp inherited from the human demonstration, but instead allows embodiment- and configuration-specific grasp strategies to emerge during policy learning.

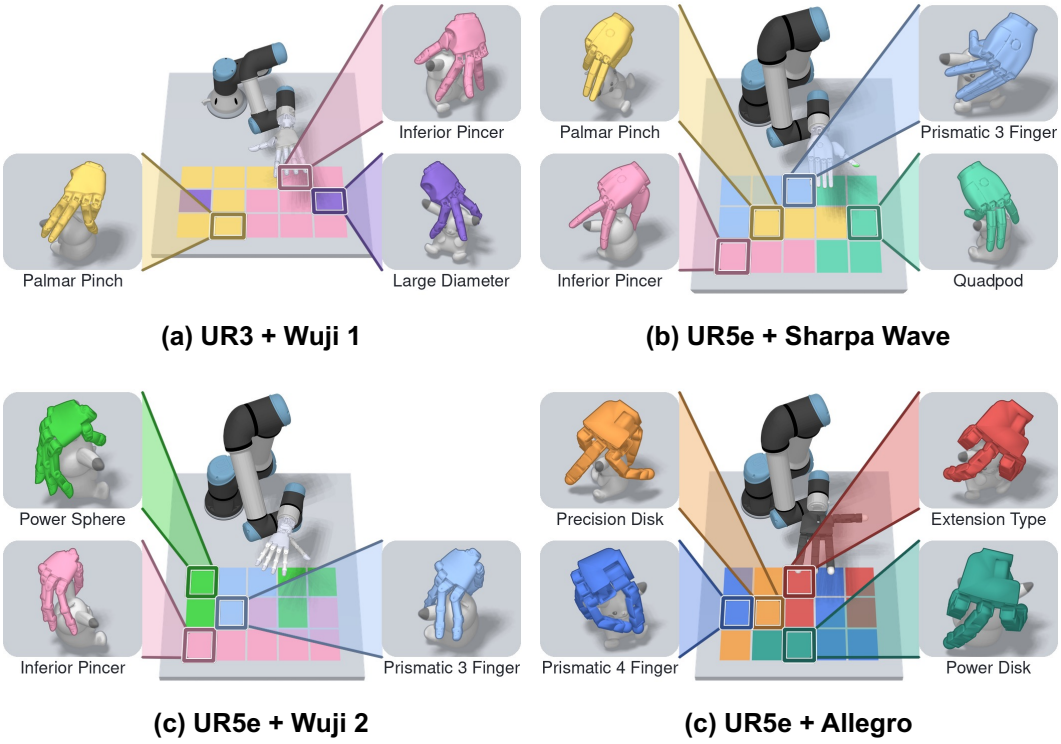


Figure 19: **Diverse grasp strategies for the Doll task.** Learned grasp types vary across dexterous embodiments and object configurations.

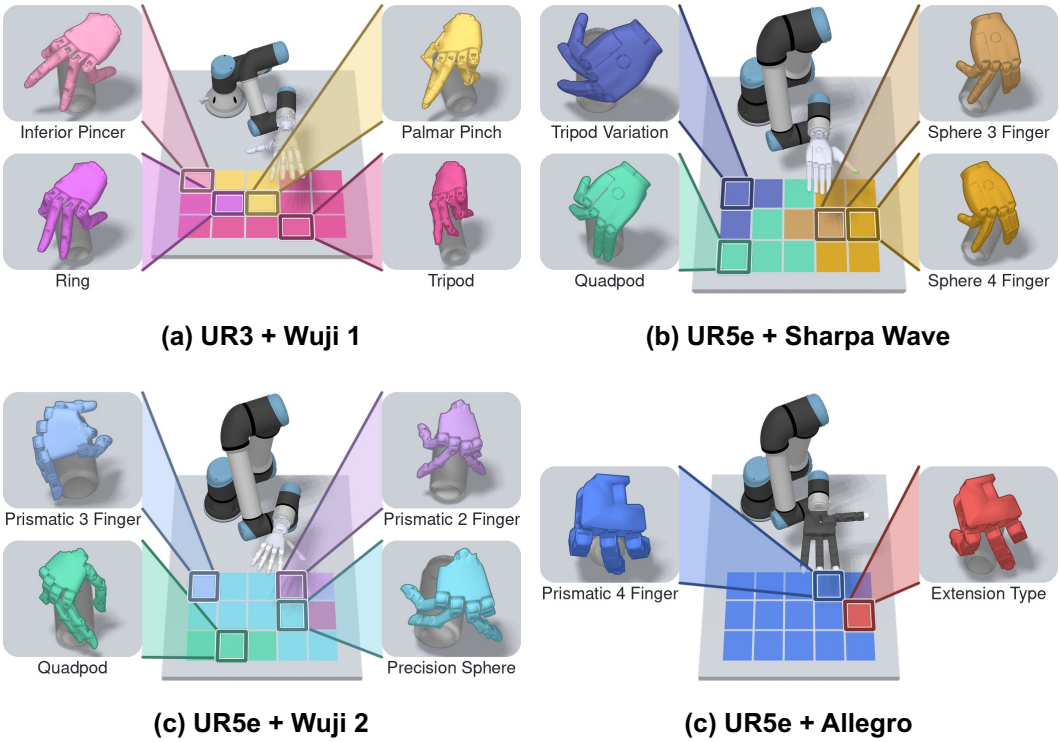


Figure 20: **Diverse grasp strategies for the Can task.** Learned grasp types vary across dexterous embodiments and object configurations.

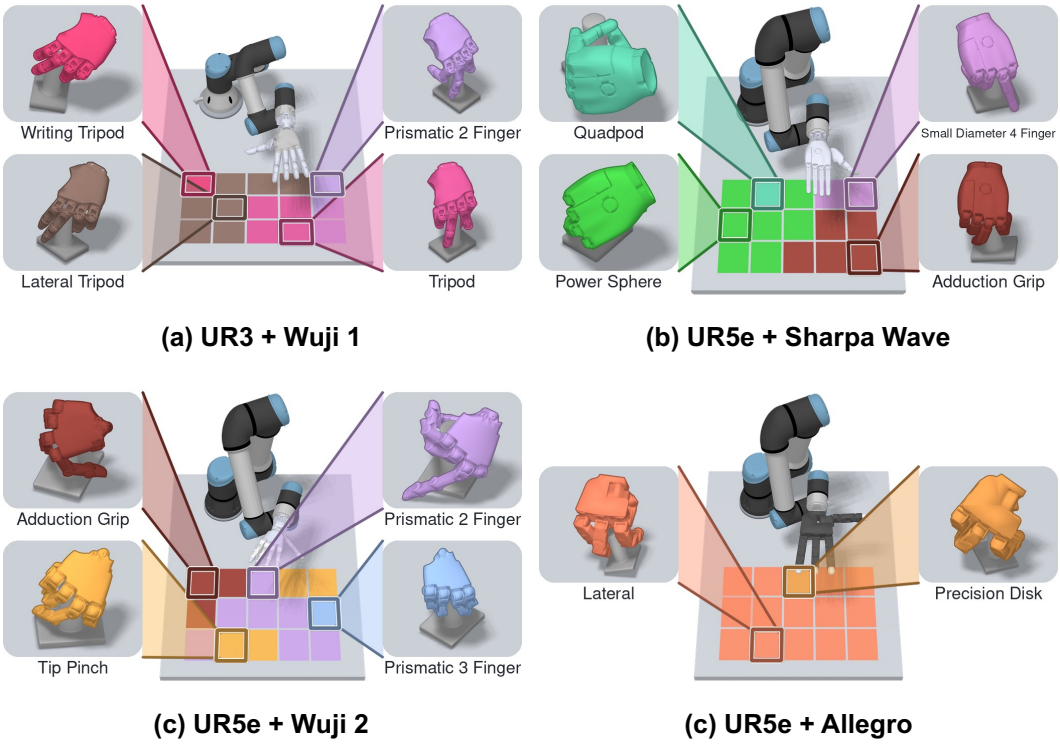


Figure 21: **Diverse grasp strategies for the Stamp task.** Learned grasp types vary across dexterous embodiments and object configurations.

E.3 Simulation Rollouts Across Diverse Dexterous Embodiments

To complement the quantitative evaluation in Table 1, we visualize successful simulation rollouts across four dexterous embodiments for all five tasks. For each embodiment and task, we show three rollouts initialized from distinct task configurations. Each row corresponds to one configuration, and the columns show the execution over time from left to right. Figure 22–26 show that separately trained policies execute the same task configuration across embodiments with different kinematics and under diverse configurations.



Figure 22: **Successful simulation rollouts across diverse dexterous embodiments for the Doll task.** For each of four dexterous embodiments, we show three successful rollouts from distinct task configurations. Each row corresponds to one configuration, and columns show the execution over time from left to right.

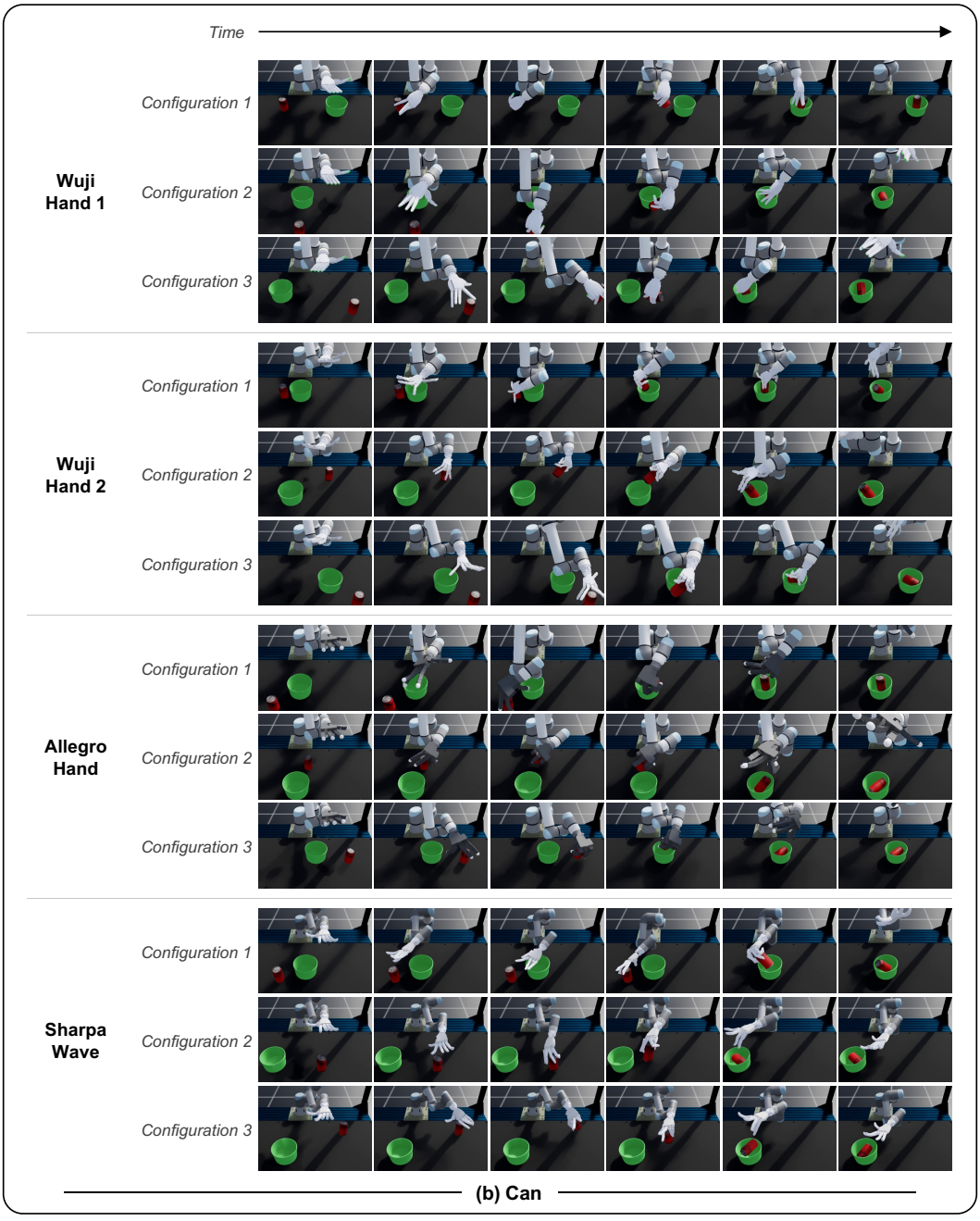


Figure 23: **Successful simulation rollouts across diverse dexterous embodiments for the Can task.** For each of four dexterous embodiments, we show three successful rollouts from distinct task configurations. Each row corresponds to one configuration, and columns show the execution over time from left to right.

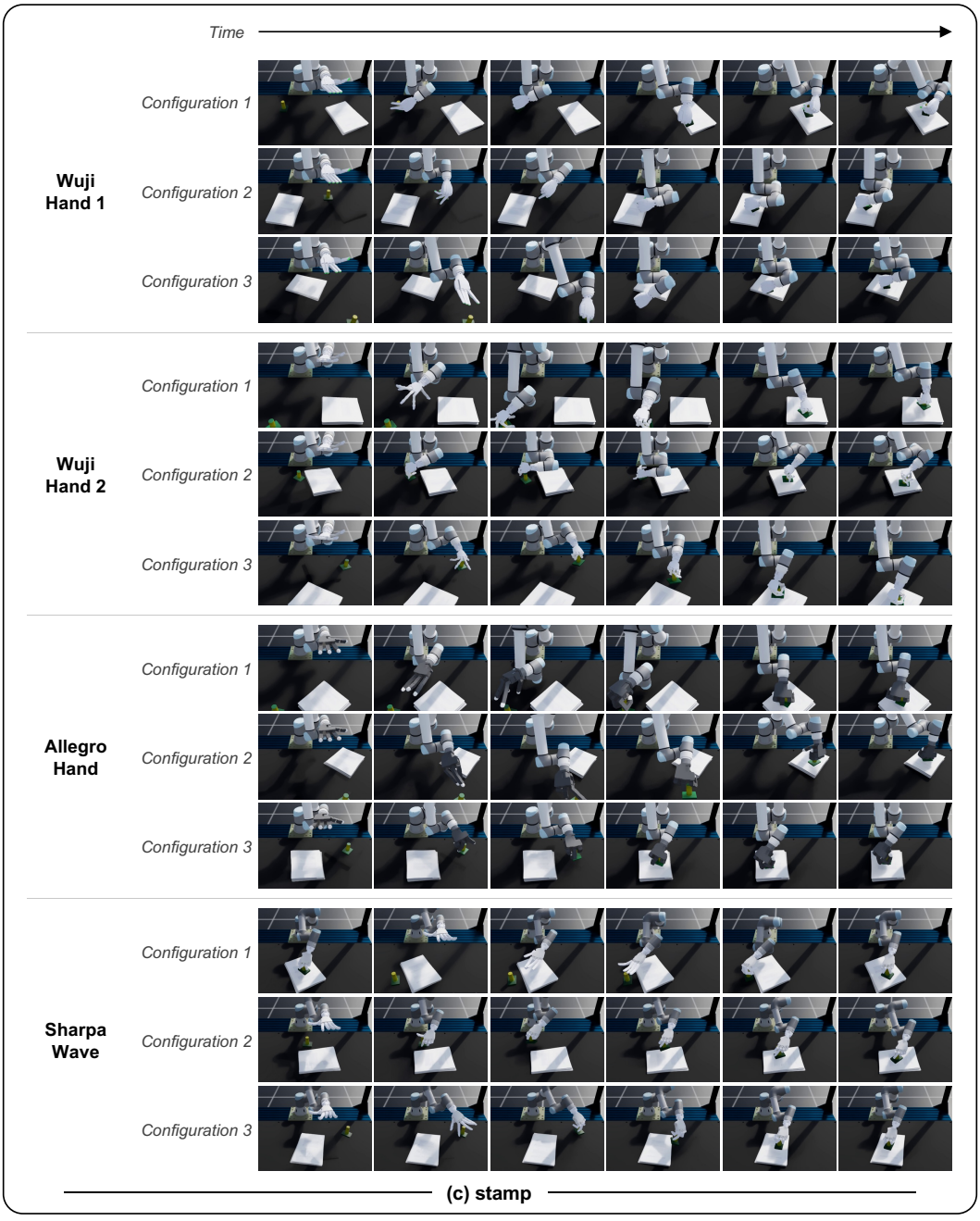


Figure 24: **Successful simulation rollouts across diverse dexterous embodiments for the Stamp task.** For each of four dexterous embodiments, we show three successful rollouts from distinct task configurations. Each row corresponds to one configuration, and columns show the execution over time from left to right.

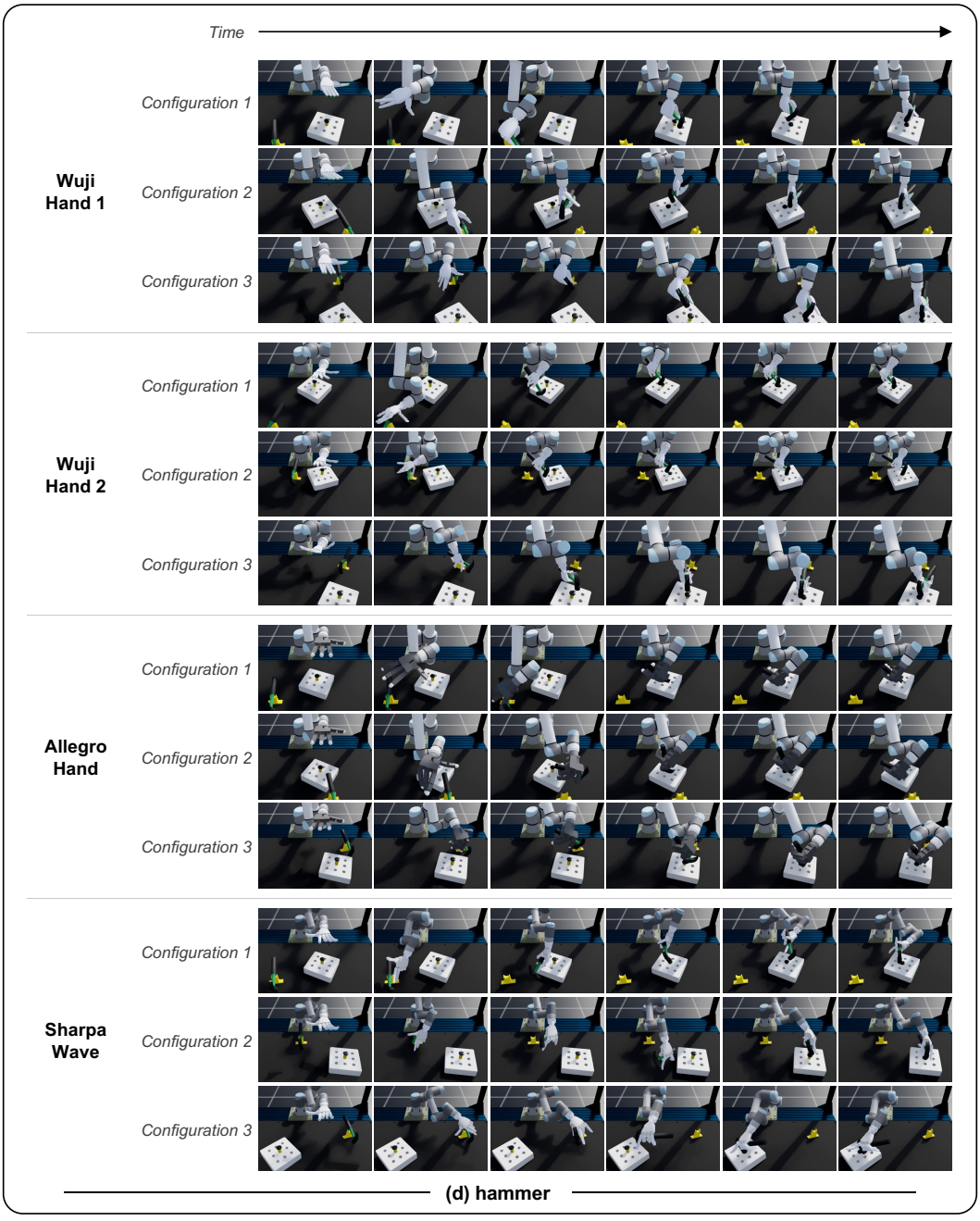


Figure 25: **Successful simulation rollouts across diverse dexterous embodiments for the Hammer task.** For each of four dexterous embodiments, we show three successful rollouts from distinct task configurations. Each row corresponds to one configuration, and columns show the execution over time from left to right.

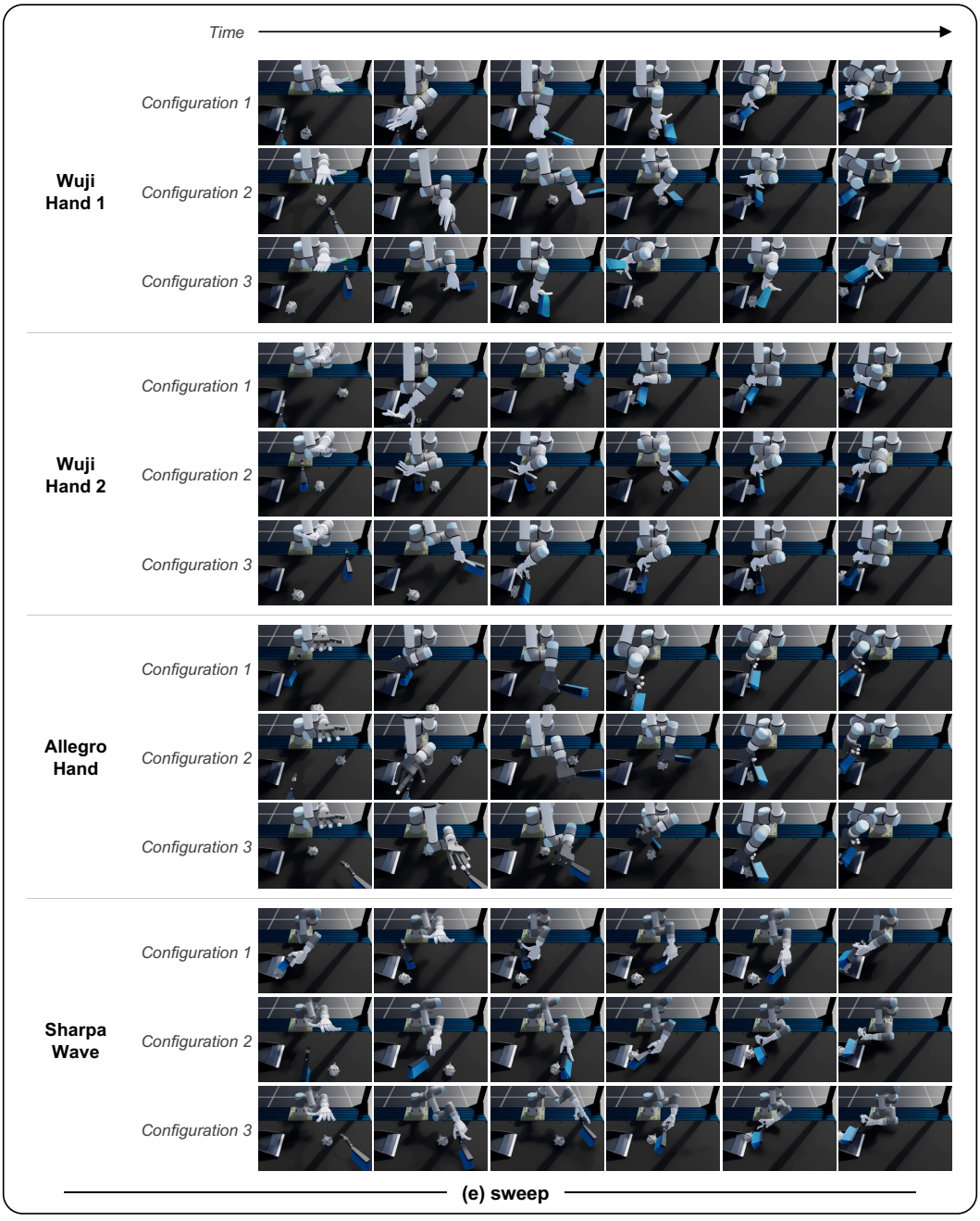


Figure 26: **Successful simulation rollouts across diverse dexterous embodiments for the Sweep task.** For each of four dexterous embodiments, we show three successful rollouts from distinct task configurations. Each row corresponds to one configuration, and columns show the execution over time from left to right.