

GOTT: Object-centric Dexterous Manipulation with a Reusable Cross-Embodiment Primitive

Yulin Liu^{1,2}, Lai Wei^{1,3}, Yen-Jen Wang^{1,4}, Akash Sharma¹, Pieter Abbeel^{1,4}, Henrik I. Christensen² and Haozhi Qi^{1,3}

¹Amazon FAR, ²UC San Diego, ³University of Chicago, ⁴UC Berkeley

<https://dex-gott.github.io>

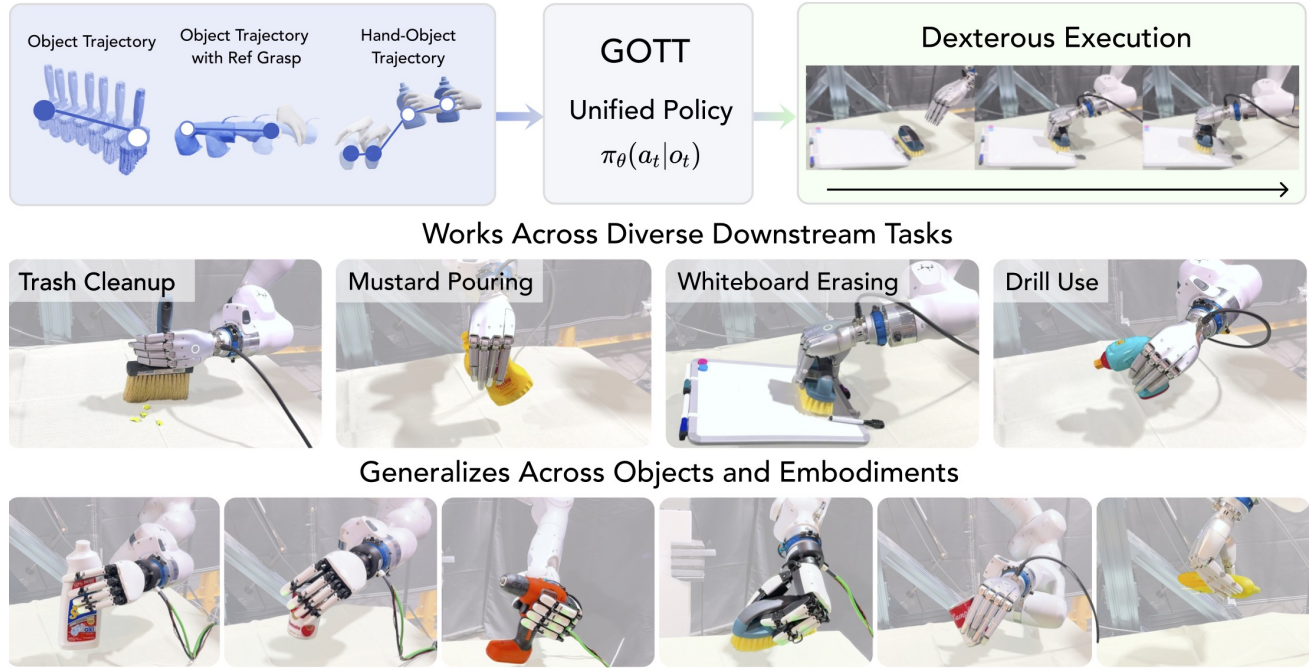


Figure 1: GOTT converts object trajectories into dexterous robot execution, with or without additional guidance from reference grasps or hand-object trajectories. The same contact-acquisition policy supports object manipulation across hand embodiments and diverse downstream tasks.

Abstract. Foundation models and large-scale human data provide rich sources of manipulation intent, but translating this intent into multi-fingered robot behavior remains difficult. Dexterous hands still lack a **reusable** low-level primitive that reliably establishes contact across tasks and embodiments. We propose GOTT, a *reach-acquire-move* framework built around a single cross-embodiment contact-acquisition primitive. Given a robot-agnostic object trajectory and a reach specification, GOTT first brings the hand near a task-relevant contact region. The shared closed-loop primitive then establishes stable contact from this approximate initialization, and a pose-conditioned controller tracks the desired object motion. Reach specifications may come from future-aware planning, external models, or human demonstrations, while the primitive and tracking backend remain unchanged. Simulation and real-world experiments show that GOTT is able to establish robust contact across diverse objects, arm-hand platforms, and seen and unseen hand morphologies. It also consistently improves end-to-end task success over open-loop grasp execution.

1. Introduction

Foundation models [16, 39] and large-scale human data [3, 60] provide rich sources of manipulation intent. For parallel-jaw grippers, such intent can be translated into robot actions through mature low-level primitives such as grasping, placing, and trajectory following [23, 24, 31]. However, the same approach does not directly extend to dexterous hands, which lack a comparable set of general-purpose low-level skills. These skills must establish complex contact configurations, coordinate high-dimensional actions, and adapt to different hand morphologies.

Existing dexterous manipulation methods usually address these challenges through task-specific sim-to-real learning [40] or hand-motion retargeting [30]. However, these approaches are not well suited to converting high-level intent into robot behavior. Task-specific policies [9, 19, 44, 45] require retraining for each new behavior, making them difficult to scale. Retargeting recovers kinematic targets from demonstrations [37], often without incorporating perceptual feedback [41]. This lack of feedback makes contact-rich tasks difficult to execute in the real world. This raises a key question: *what is the right interface for translating high-level manipulation intent into low-level dexterous execution?*

Object-centric motion representations provide a promising answer to this question. Prior work has shown that many rigid-object manipulation tasks can be described by the desired object motion [20, 27, 43], abstracting away embodiment-specific robot motion. However, an object trajectory specifies *how the object should move*, but not *where and how the hand should establish contact*. This distinction is critical for functional manipulation: a geometrically stable grasp may still be unsuitable for pouring, sweeping, or tool use. Prior methods address this gap by supplying a pre-grasp or hand reference [10, 32], or by learning contact acquisition jointly with trajectory execution [26, 28]. While effective, these designs do not expose contact acquisition as a reusable skill across high-level sources and robot embodiments.

We propose **General Object Trajectory Tracking** (GOTT), a framework for executing diverse object motions across tasks and robot embodiments using a shared contact-acquisition primitive. We represent high-level manipulation intent using a desired object trajectory and a *reach specification*, which specifies how to bring the hand near a task-relevant contact region. These components may come from different sources. For example, the object trajectory may be extracted from a video, while the reach specification may be generated through affordance-aware planning or obtained by retargeting a human trajectory.

GOTT implements this interface through a *reach*, *acquire*, and *move* decomposition. During *reach*, the robot follows the reach specification toward the selected contact region. During *acquire*, a learned closed-loop contact-acquisition primitive establishes stable contact from this approximate initialization. During *move*, an object-pose-conditioned controller tracks the desired object trajectory. The primitive uses only local hand-object geometry and proprioceptive history, while a canonical per-link representation allows the same policy to be shared across hands. Reach specifications can come from VLM-guided, future-aware planning, external models, or demonstrations, while the contact-acquisition primitive remains unchanged.

We evaluate GOTT in simulation and the real world. First, we test whether the contact-acquisition primitive can robustly grasp diverse objects across different hands and transfer zero-shot to unseen hand morphologies. We then evaluate the full pipeline on object-trajectory tracking tasks, including trash cleanup, pouring, whiteboard erasing, and tool use. We compare against end-to-end trajectory learning, showing that our decomposition enables robustness to workspace shifts and reuse across robot arms. We also show that the same primitive supports high-level specifications from generated videos, keyframes, human demonstrations, and hand-object interaction datasets. Finally, we conduct ablations on the primitive’s observation and reward design and the system’s planning modules.

2. Related Work

High-level task interfaces. Foundation models and human demonstrations can specify robot manipulation tasks through language, visual constraints [24], code [31], or generated videos [43]. ManipGen [12] combines reusable local policies with foundation models and motion planning. Another line of work learns from human videos by extracting intermediate motion representations, such as point trajectories, object flow, or object-centric motion plans [4, 20, 29, 56, 58, 65]. SPOT [21] learns object-pose trajectory plans from human demonstrations and uses them to condition a robot policy. VLM-Scaffold [13] adapts a similar pipeline to dexterous hands but relies on task-specific residual RL, making it challenging to scale across tasks and embodiments. LUCID [18] learns embodiment-agnostic intent from unstructured human videos and connects it to embodiment-specific sensorimotor policies through object flow and a rough palm pose. In contrast, GOTT represents high-level intent using an object trajectory and a flexible reach specification, and connects these inputs to robot execution through a reusable dexterous contact-acquisition primitive.

Trajectory-based dexterous manipulation. D-Grasp [11] combines reference-conditioned grasp control with global motion synthesis to move objects to target poses. Methods that start from a reference trajectory typically follow one of two strategies. The first performs trajectory-specific transfer, converting a human or object trajectory into robot motion through kinematic optimization, sampling [41], or reinforcement learning [30, 37]. Human2Sim2Robot [35] uses a single object trajectory and a pre-grasp to learn a task-specific robot policy. ManipTrans [30] and REGRIND [15] transfer hand-object interaction trajectories through residual learning, while SPIDER [41] performs fast physics-informed retargeting. These transfer methods can reproduce a given demonstration but are often tied to a specific trajectory or embodiment.

The second strategy trains policies conditioned on multiple reference trajectories. ObjDex [10] and DexTrack [32] condition policies on object or hand trajectories. AdaDexTrack [1] adds online modulation to improve tracking of noisy references. OmniGrasp [36] learns to grasp diverse objects and follow object trajectories with a simulated humanoid. SimToolReal [26] and Dex4D [28] learn general object-centric policies conditioned on target object poses or point tracks, jointly handling contact acquisition and motion execution. Grasp to Act [17] combines human-guided grasp optimization with residual RL for online finger adaptation. In contrast, GOTT factors execution into *reach*, *acquire*, and *move*: high-level inputs specify the desired object motion and a task-relevant reach, while a trajectory-agnostic primitive establishes contact before trajectory tracking begins. This decomposition enables the same contact-acquisition primitive to be reused across trajectories, objects, and robot embodiments.

Dexterous grasping. Existing learning-based methods develop generalizable tabletop grasping policies [34, 50, 59, 63]. Grasp synthesis and evaluation methods instead generate or refine grasp targets [7, 8, 14, 33, 51, 53]. GraspXL [62] and ConGraspXL [64] condition grasping motions on approach, contact-region, or pose constraints. Functional grasping methods use affordances, human grasp references, or language to select task-relevant grasps [2, 22, 52]. Cross-embodiment methods further share grasping policies or representations across hands: CrossDex [61] uses a unified observation space and retargeted eigengrasp actions, while OHRA [54] uses canonical hand representations and UniMorphGrasp [57] conditions grasp generation on morphology. GET-Zero [42] studies zero-shot transfer to unseen hand structures for in-hand rotation. These works advance grasping and cross-hand transfer, while primarily focusing on grasp acquisition itself. In contrast, GOTT treats contact acquisition as a reusable intermediate primitive that refines an approximate, task-relevant reach into stable contact before downstream object-trajectory tracking. This enables the same primitive to be reused across objects, tasks, trajectories, and hand embodiments.

3. Dexterous Manipulation via General Object Trajectory Tracking

We propose GOTT, a *reach-acquire-move* framework (Section 3.1) for executing rigid-object motions with dexterous hands. Its high-level interface consists of a desired object trajectory τ_o and a reach specification ρ that brings the hand near a task-relevant contact region. At its core, a reusable cross-embodiment contact-acquisition primitive grounds this interface by turning an approximate reach into stable contact (Section 3.2). In the object-centric setting, a future-aware planner constructs ρ from the object mesh, desired trajectory, and task description (Section 3.3). Alternatively, external models or human demonstrations can provide ρ (Section 3.4), while the primitive and tracking backend remain unchanged (Section 3.5).

3.1. Reach-Acquire-Move Decomposition

Given a desired object trajectory $\tau_o = \{T_{O,t}\}_{t=0}^T$ where $T_{O,t}$ denotes the object pose at timestep t , and a reach specification $\rho = (\xi_{\text{app}}, g_{\text{init}})$, GOTT decomposes dexterous execution into three stages. During *reach*, the robot follows the approach trajectory ξ_{app} to an approximate initial configuration g_{init} . During *acquire*, the contact-acquisition primitive uses closed-loop feedback to establish stable contact. During *move*, the pose-conditioned tracker moves the object along τ_o .

This decomposition separates high-level reasoning about functional contact selection, collision-free approach, and long-horizon reachability from local contact-rich control. Consequently, the high-level source needs to provide only a feasible approach to an approximate initial configuration, rather than an exactly executable final grasp.

3.2. General-Purpose Contact Acquisition Primitive

Establishing stable contact from an initial hand configuration near the object remains challenging. State-of-the-art grasp-generation methods either execute grasps open loop [7, 53], do not account for the functional suitability of the selected initial grasp [34, 63], or do not generalize across embodiments. We therefore train a local, closed-loop contact-acquisition policy using reinforcement learning in simulation and transfer it to the real world. The policy uses local hand-object geometry and proprioceptive history, while a canonical representation enables sharing across hands. This design allows the same primitive to be reused across tasks, high-level sources, objects, and hand embodiments. We first describe the single-hand control policy and then its extension to cross-embodiment control.

Environment setup. We select 16 YCB objects [5] as our training set and collect several stable resting poses for each object. For each object–pose pair, we use BODex to generate 4,000 initial hand configurations of the type described in Section 3.3 and shown in Figure 2A. Each episode starts from one of these initial hand configurations. The policy must establish stable contact and lift the object.

Observations and actions. The observation contains three components: (1) the link-distance representation from [63], in which each hand link is represented by its displacement to the nearest point in the object point cloud; (2) hand-link positions; and (3) joint positions and previous joint targets. Components (2) and (3) are stacked over the five most recent timesteps. The policy outputs a 6D wrist-delta command and a vector of hand-joint deltas: $a_t = \pi_\theta(o_t)$.

Rewards. Our reward function is defined as: $r = w_d r_{\text{dist}} + w_c r_{\text{contact}} + w_l r_{\text{lift}} + w_o r_{\text{obj}}$. The four terms encourage link-object proximity, opposing contact between the thumb and another finger, lifting after contact, and object stability, respectively. Additional definitions and weights are provided in the appendix.

Domain randomization. During training, we randomize object scale, friction, and mass; robot-link

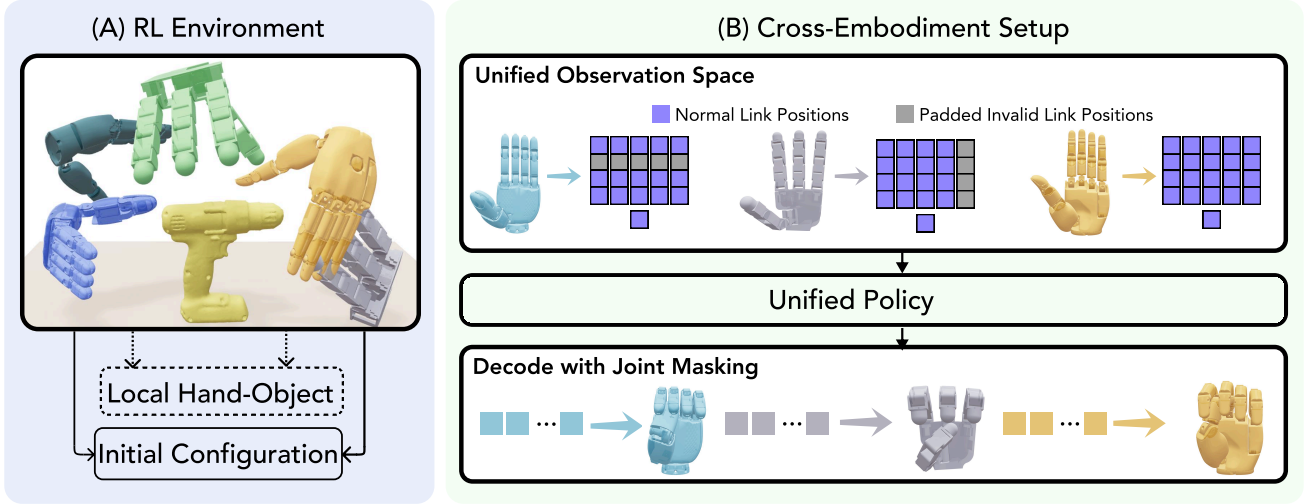


Figure 2: Contact-acquisition training and cross-embodiment policy. (A) The RL policy starts from randomized initial configurations and learns to establish stable contact using local hand-object observations. (B) Padding and binary masks map different hands into a shared observation and action space, allowing one policy to operate across hand embodiments.

friction; and actuator stiffness and damping. We also apply random force and torque perturbations once the object is held to improve grasp robustness.

Contact prediction. During deployment, we use a proprioception-based contact predictor to determine when to switch from acquisition to tracking. Because proprioceptive and action histories can reveal contact events [44], we train a separate network after the policy converges: $\hat{c}_t = f_\phi(h_{t-k+1:t}^{\text{prop}}) \in [0, 1]^5$, where $h_{t-k+1:t}^{\text{prop}}$ contains recent joint positions, joint targets, and hand-link positions, and $\hat{c}_t$ contains the per-finger contact probabilities. We train the network in simulation using ground-truth per-finger contact labels. Acquisition terminates when the predicted contact probabilities for the thumb and at least one other finger exceed τ_c for k consecutive timesteps.

Unified cross-embodiment representation. To train one controller across multiple multi-fingered hands, we map hand-specific observations and actions into a canonical representation, as shown in Figure 2B. For link-level observations, we map each hand onto a 21-keypoint topology consisting of one palm keypoint and four keypoints per finger. Missing keypoints are filled using the nearest available links, and a binary link mask distinguishes valid entries from padded ones. Joint positions and hand actions are similarly mapped to a fixed 26-slot canonical joint order. Missing joint slots are zero-padded, and a binary joint mask identifies the active slots. The policy outputs a 6D end-effector delta and a canonical joint-delta vector. Before execution, padded joints are masked out, and the remaining entries are remapped to the robot’s native joint indices. This representation allows one policy to operate across hands with different joint counts and kinematic structures without hand-specific encoder or decoder networks.

3.3. Reach Specifications from Future-Aware Planning

The reach stage can be instantiated from different high-level inputs. We first consider the default object-centric setting, in which the inputs are an object mesh $\mathcal{M}$, a desired object trajectory τ_o , and a task description. The planner must determine where to contact the object and select an initial hand configuration that allows a feasible approach and downstream arm motion. Existing geometry-based grasp synthesis methods [7, 53] can generate locally stable candidates but do not consider task affordances

or the full object trajectory. A candidate may therefore be functionally unsuitable, or it may be initially reachable but become infeasible later because of joint limits or collisions (Figure 3). We address these issues through VLM-based affordance reasoning and future-aware IK selection to select a task-relevant initial configuration for which the induced arm motion remains feasible along τ_o .

VLM-based affordance reasoning. Geometric stability alone does not determine whether a grasp is suitable for the task. For example, grasping a drill by its chuck may be stable but does not support functional tool use. We therefore use a VLM to identify a task-relevant contact region. We render the object from four viewpoints and prompt a VLM to identify the task-relevant grasping part and predict keypoints spanning its extent. The selected 2D keypoints are back-projected onto the mesh and aggregated across views by voting, yielding a task-relevant contact region $\mathcal{R} \subset \partial\mathcal{M}$.

Region-conditioned candidate generation. We use BODex’s lightweight initialization stage [7] to sample candidate hand poses and joint configurations from object geometry within $\mathcal{R}$, yielding $\mathcal{G} = \{g_i\}_{i=1}^N$ ($N = 10$). Each g_i is an approximate initialization rather than a final grasp. The learned contact-acquisition primitive then establishes contact more robustly, as shown in our experiments.

Trajectory-aware selection. For each g_i , we first plan a collision-free approach trajectory ξ_{app} from the robot’s current configuration to the candidate initialization. We then roll out the nominal hand motion induced by keeping the hand-object transform fixed while the object follows τ_o . We then solve inverse kinematics (IK) along this motion and discard candidates for which the approach is infeasible, the IK solutions violate joint limits, or the solver fails to converge. Among the remaining candidates, we select g^* with the least cumulative joint motion: $g^* = \arg \min_{g_i \in \mathcal{G}_{\text{valid}}} \sum_{t=1}^T \|q_{i,t}^{\text{arm}} - q_{i,t-1}^{\text{arm}}\|_2$, where $\mathcal{G}_{\text{valid}}$ is the set of remaining candidates and $q_{i,t}^{\text{arm}}$ is the arm IK solution for candidate g_i at timestep t . Implementation details are provided in the Appendix B.

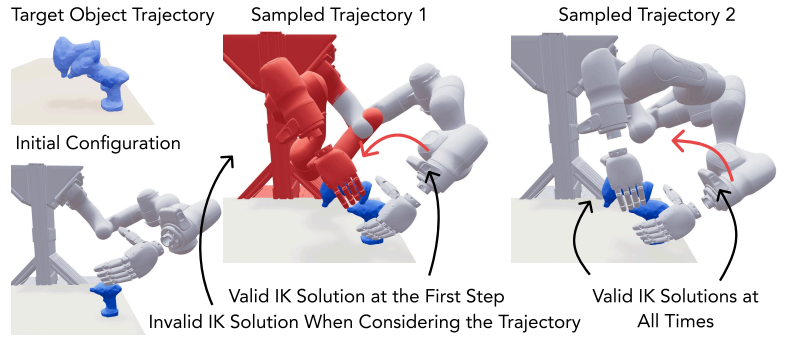


Figure 3: Future-aware planning. We roll out the hand motion induced by each candidate initial configuration over the full object trajectory and solve IK along this motion. Red segments indicate joint jumps from discontinuous IK solutions. We reject candidates that are initially reachable but become infeasible later and select the remaining candidate with the least cumulative joint motion.

3.4. Reach Specifications from Human Guidance and External Models

Section 3.3 constructs the reach specification ρ from object-side information. Alternatively, human demonstrations or external models can supply contact regions, initial hand configurations, or approach trajectories directly, replacing the corresponding planning steps (Figure 4). The resulting reach specification is executed by the same contact-acquisition primitive and trajectory tracker.

Contact regions or initial configurations. External models or human demonstrations can provide a task-relevant contact region or an initial hand configuration. Such a contact region replaces VLM-based affordance reasoning, while an initial configuration also bypasses candidate generation. The remaining planning steps construct a feasible approach and check downstream arm feasibility to form ρ .

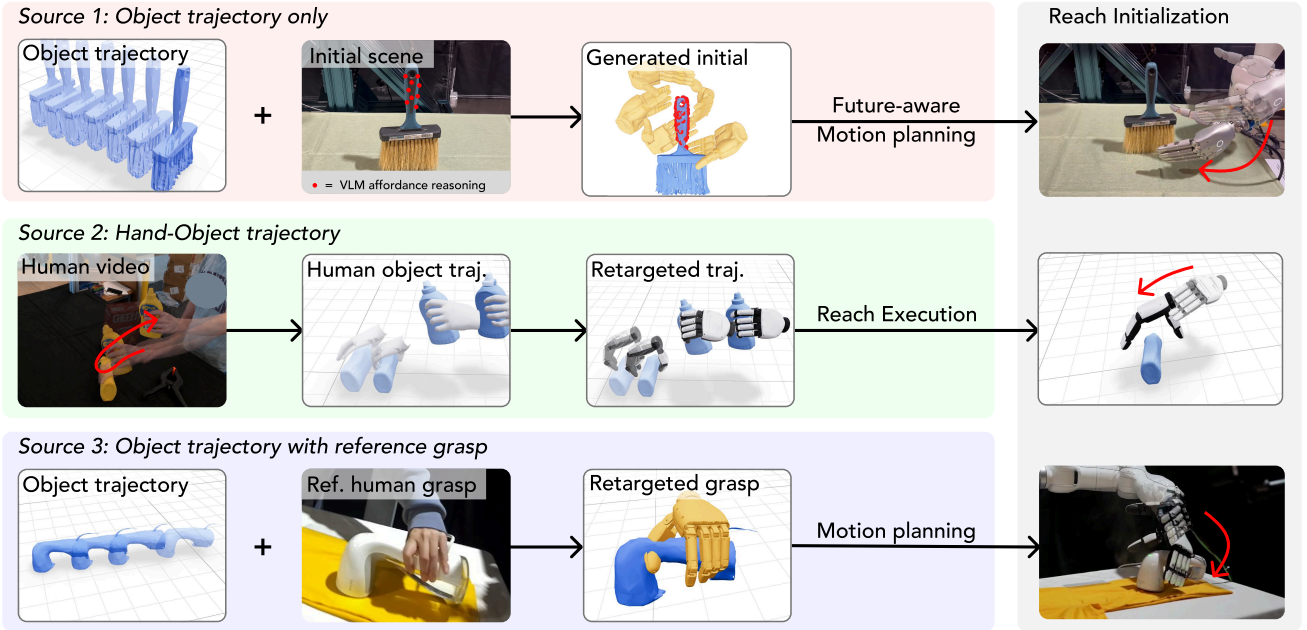


Figure 4: Alternative sources of reach specifications. Given a desired object trajectory, GOTT can construct a reach specification through future-aware planning or use additional guidance from reference grasps or human approach trajectories. The shared contact-acquisition primitive refines the resulting approximate initializations into stable contact.

Retargeted human approaches. Human-object datasets such as DexYCB [6] and OakInk [60] provide synchronized hand and object motions. We retarget the human approach segment to the robot hand to obtain ξ_{app} and its terminal configuration g_{init} , which together define ρ . This supplied approach replaces approach generation. Its endpoint need not be an exact final grasp: the contact-acquisition primitive establishes stable contact, after which the trajectory tracker follows τ_o .

Richer guidance replaces more planning steps, but may not account for robot-specific reachability or collisions. We therefore retain future-aware planning as the default for arm-hand execution. Qualitative examples of these alternative compositions are available on the project website.

3.5. Execution from High-Level Specifications

Regardless of the high-level source, the execution backend receives a desired object trajectory τ_o and a reach specification ρ . During real-world deployment, we use SAM2 [46] to segment the target object and FoundationPose [55] to estimate and track its pose. GOTT executes the task in three stages: *reach*, *acquire*, and *move*.

Reach. The robot follows the approach trajectory specified by ρ . The system switches to the contact-acquisition primitive once the minimum distance between the hand links and the object surface falls below $\epsilon_{\text{approach}}$.

Contact acquisition. At this transition, the contact-acquisition primitive takes control of the wrist and fingers. It locally adjusts the hand to compensate for object pose errors, geometry mismatch, and contact uncertainty. During this phase, we run the proprioception-based contact predictor described in Section 3.2. Once the predictor detects stable contact, the system proceeds to the move stage.

Pose-conditioned end-effector tracking. After stable contact is detected, the robot performs a short lift before tracking the desired object trajectory. We approximate the hand-object transform as fixed during tracking. At each timestep, we compute an end-effector target from the desired object pose and the estimated hand-object transform, correct the target using the observed object pose error, and send it to the low-level position controller.

We evaluate the contact-acquisition primitive’s robustness, cross-embodiment transfer, and contribution to end-to-end task success. We further examine its reuse across workspaces and with different high-level sources, and evaluate key design choices through ablations.

(a) Test objects

(b) Robot platforms

We obtain object meshes using AR Code¹ on an iPhone and use them for planning and pose tracking. During execution, SAM2 [46] segments the target object in the initial frame, and FoundationPose [55] estimates and tracks its 6D pose using the mesh and segmentation mask. FoundationPose runs at 45 Hz, while the policy runs at 10 Hz and uses the latest pose estimate at each control step.

Simulation setup. We train a unified policy for XHand, Sharpa, and Allegro in IsaacLab [38] using PPO [48]. We train for 2,000 epochs using 4,096 parallel environments. The training set includes 16 YCB objects [5], with 4,000 initial configurations generated using BODex for each object–resting-pose pair.

8

4.1. Is It a Reliable General-Purpose Contact Primitive?

We isolate the low-level contact-acquisition primitive by evaluating object lifting in simulation and the real world, without full-arm planning.

4.1.1. Simulation grasping evaluation

We use a floating-hand tabletop setup and evaluate on DexGraspNet objects [51], following the pre-processing protocol of BODex [7]. After filtering out flat objects and rescaling the remaining objects to four sizes, the evaluation set contains 9,588 assets and 54,149 object-scale-pose instances. We run simulations in IsaacLab with the friction coefficient set to 1.5 and the object mass set to 100 g. All physics parameters are fixed across evaluations.

Baselines. We compare against two representative grasp synthesis methods. BODex [7] optimizes grasp poses for each robot-object pair using bilevel optimization. DRO [53] is a cross-embodiment grasp prediction network, for which we use the authors’ released checkpoints. Since both methods output grasp targets rather than closed-loop acquisition policies, we execute their outputs open-loop using their original execution controllers.

Evaluation protocol. For each object-scale-pose instance, we generate 20 initial poses using BODex’s first stage (Section 3.3) and share them across methods. BODex refines them via bilevel optimization, DRO uses them to specify initial approach directions, and our policy starts directly from them for closed-loop contact acquisition. In the same tabletop setup, BODex and DRO execute open-loop from predicted pre-grasps, effectively assuming successful motion planning to these poses. Our method additionally handles the approach from the raw initial poses.

Metrics. We report lifting success rates across hand embodiments, objects, and initial object poses. A trial is successful if the hand establishes contact, lifts the object by at least 5 cm, and keeps it stable for the full evaluation horizon.

Results. The results are shown in Table 1. GOTT consistently outperforms BODex and DRO in lifting success across all three hands. Notably, although the policy is trained on only 16 YCB objects, it generalizes to the full evaluation set of 9,588 DexGraspNet assets.

The success rates for BODex and DRO are lower than those reported in their original papers due to our stricter tabletop protocol. Prior work initializes objects without gravity and applies gravity only after contact is established, whereas we place objects on a table with gravity enabled from the start and execute directly. Appendix E details the baseline implementations and evaluation protocols and presents controlled comparisons.

4.1.2. Real-world grasping experiments

We evaluate the contact-acquisition primitive using the same checkpoint on three arm-hand platforms: Franka-Sharpa, Franka-XHand, and Tianji-XHand. This tests both cross-hand generalization and transfer across arm platforms through the end-effector control interface. The evaluation setup is shown in Figure 5. We run 5 trials on each of 10 objects, for 50 rollouts per platform. A trial is successful if the hand establishes contact, lifts the object by at least 5 cm, and keeps it stable for at least 3 seconds. GOTT achieves success rates of 90%, 90%, and 96% on Franka-Sharpa, Franka-XHand, and Tianji-XHand, respectively. The slightly lower success rates on Franka are mainly due to the more constrained reachable workspace in our Franka setup.

Table 2: **Zero-shot generalization to unseen Allegro variants.** Lifting success rate (%), reported as mean $\pm$ standard deviation over three seeds. xyzw denotes per-finger link counts (4444 = original Allegro). Oracle trains one policy per variant; Allegro-only trains on the original Allegro; Three-hand is our unified policy over XHand+Sharpa+original Allegro, evaluated zero-shot.

Variant	4433	4333	4324	Avg.
Oracle (per-variant)	69.0 $\pm$ 6.8	71.6 $\pm$ 1.1	76.0 $\pm$ 4.0	72.2 $\pm$ 1.2
Allegro-only (zero-shot)	29.4 $\pm$ 19.7	12.9 $\pm$ 8.6	9.4 $\pm$ 8.1	17.2 $\pm$ 12.1
Three-hand (zero-shot)	74.4 $\pm$ 6.5	72.2 $\pm$ 4.8	69.6 $\pm$ 9.6	72.1 $\pm$ 6.9

4.2. Does It Generalize across and beyond Training Hands?

4.2.1. Cross-hand evaluation and representation comparison

The lower portion of Table 1 compares GOTT with CrossDex [61], an RL baseline that uses a shared action space through human-hand retargeting. To isolate the effect of the observation–action representation, we reimplement CrossDex within our framework. We adopt the observation and action design from its released code while keeping the reward, PPO hyperparameters, and training and evaluation protocol the same as those used for GOTT. Specialist policies are trained separately for each hand, while unified policies are trained jointly on XHand, Sharpa, and Allegro. Our unified policy with the per-link canonical representation achieves 78.7% average success versus 51.7% for CrossDex. Joint training also improves average success over our specialist policies, supporting positive transfer across morphologies.

4.2.2. Zero-shot transfer to unseen Allegro variants

To further probe cross-embodiment generalization, we test unseen Allegro variants generated by removing finger links. We evaluate our three-hand unified policy, trained on XHand, Sharpa, and the original Allegro hand, *zero-shot* on these variants. We compare it with (i) per-variant oracle policies trained directly on each variant and (ii) a policy trained only on the original Allegro hand. As shown in Table 2, our three-hand policy averages 72.1% success across the three variants, matching the average of the per-variant oracle policies (72.2%), whereas the Allegro-only policy achieves 17.2%. Transfer therefore comes from multi-morphology training, not the canonical representation alone. Figure 6 further shows that the policy produces coordinated grasp configurations on these unseen variants.

4.3. Does the Primitive Improve End-to-End Task Success?

We evaluate whether the contact-acquisition primitive improves end-to-end execution on four real-world manipulation tasks using the Franka-Sharpa platform. For each task, the robot must approach the object, acquire stable contact, and move the object along the specified trajectory. We compare GOTT with two matched open-loop variants. Each replaces our region-conditioned candidate generation with the full grasp generation procedure of BODex [7] or DRO [53] and executes the predicted grasp open-loop. All other components, including perception, VLM-based affordance reasoning, future-aware IK selection, and trajectory tracking, are kept fixed, isolating the contribution of closed-loop contact acquisition.

Real-world tasks. We evaluate *Trash Cleanup* (grasp a brush and sweep trash), *Mustard Pouring* (grasp a mustard bottle and rotate it to a pouring orientation), *Whiteboard Erasing* (grasp an eraser and sweep across the board), and *Drill Use* (grasp a power drill and rotate it to a working orientation), as illustrated in Figure 7. These tasks require both translation and rotation, and some require functional contacts, such as grasping the drill near its handle. Each task is repeated 10 times across varied initial poses.



Figure 6: Qualitative grasps on training and unseen embodiments. We compare the full GOTT (top) with the variant without r_{dist} (bottom) on three training hands and three unseen Allegro variants.

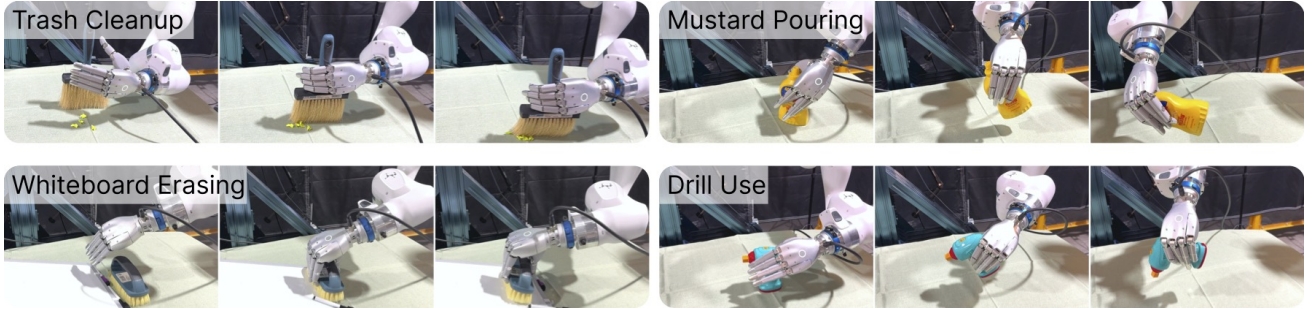


Figure 7: Qualitative illustration of four real-world tasks.

Metrics. We report grasp success rate and task success rate. Grasp success measures whether the robot acquires stable contact, while task success measures whether it completes the downstream trajectory-tracking task.

Results. Table 3 shows that GOTT achieves the highest grasp and task success rates on every task, with an average task success rate of 80.0% versus 35.0% for both baselines. Baseline failures include unsuitable contact regions, object displacement during approach, and cage-like grasps lacking firm opposing contact between the fingers and thumb. The last failure mode explains DRO’s 0/10 task success on *Trash Cleanup*: loose contact around the brush handle causes the brush to slip during sweeping. While the open-loop baselines are sensitive to planning errors, our closed-loop controller adjusts the wrist and fingers to recover from imperfect initializations.

4.4. Does Decomposition Improve Robustness to Workspace Variations?

SimToolReal [26] and Dex4D [28] jointly learn contact acquisition and object tracking with embodiment-specific policies. In contrast, GOTT uses a reusable contact primitive, arm-specific IK, and object-pose-conditioned tracking to separate contact acquisition from robot-specific reachability.

We evaluate GOTT on the eraser object from SimToolReal’s DexToolBench, using a 5 cm lifting trajectory, over 600 simulated episodes. We vary the object’s planar position and the table’s height offset z relative to the robot base (Figure 8).

Table 3: **Real-world task results.** Per-task grasp success rate (gSR) and task success rate (tSR) across four manipulation tasks on Franka-Sharpa platform, 10 trials each. Best per column in bold.

	Trash Cleanup		Mustard Pouring		Whiteboard Erasing		Drill Use		Avg. (%)	
Method	gSR	tSR	gSR	tSR	gSR	tSR	gSR	tSR	gSR	tSR
DRO	6/10	0/10	5/10	5/10	5/10	3/10	7/10	6/10	57.5	35.0
BODex	5/10	2/10	4/10	2/10	7/10	3/10	7/10	7/10	57.5	35.0
Ours	8/10	7/10	8/10	7/10	10/10	9/10	9/10	9/10	87.5	80.0

Near SimToolReal’s training configuration, its success rate is 0.983, compared with 0.792 for GOTT. Our failures are mainly due to IK singularities under a constrained end-effector orientation. Below $z = 0.4$ m, SimToolReal’s success rate drops to 0.002, while GOTT achieves 0.914. In this comparison, we reuse our contact primitive on SimToolReal’s iiwa-Sharpa platform and evaluate SimToolReal using its released checkpoint. Adapting SimToolReal to a different arm embodiment would instead require roughly seven days of training, whereas our primitive is arm-agnostic by design. These results support separating workspace-specific reachability from reusable contact control.

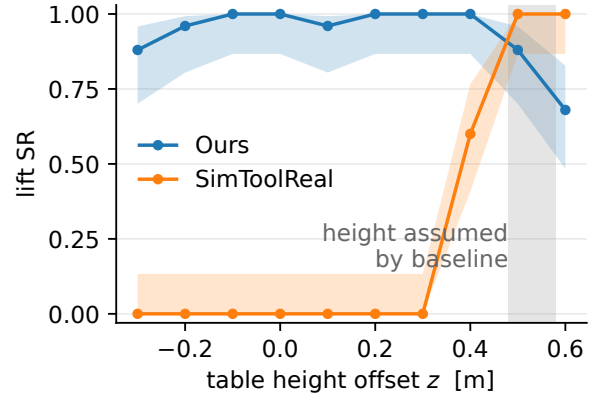


Figure 8: Success rate versus table height offset z on the SimToolReal eraser object.

4.5. Does It Support High-Level Specifications from Different Sources?

We test whether GOTT accepts high-level specifications from diverse sources. These include object trajectories obtained from different sources and unseen human demonstrations that also specify the hand’s approach. Across all sources, the contact-acquisition primitive remains fixed, without task- or trajectory-specific fine-tuning.

4.5.1. Object trajectories from diverse sources

Figure 9 shows that object trajectories can be obtained from human demonstrations, generated videos, and manually specified object-pose keypoints. For human demonstrations, we track object SE(3) trajectories using FoundationPose [55]. For generated videos, we follow RigVid [43], estimating per-frame depth with RollingDepth [25] and recovering object trajectories with FoundationPose [55]. For keypoints, we interpolate between user-specified object poses. The resulting trajectories from all three sources can be used directly by the GOTT pipeline.

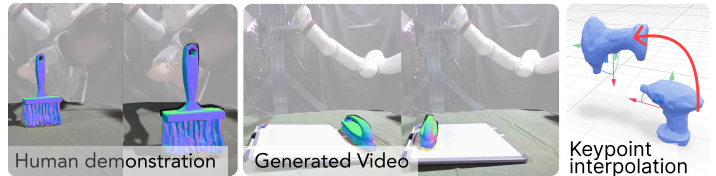


Figure 9: Visualization of three sources of object trajectories.

4.5.2. Zero-shot execution of unseen human demonstrations

We evaluate all 499 left-hand DexYCB sequences [6], each containing human-hand motion and an object trajectory. We kinematically retarget the hand motion to XHand for the approach, acquire stable contact

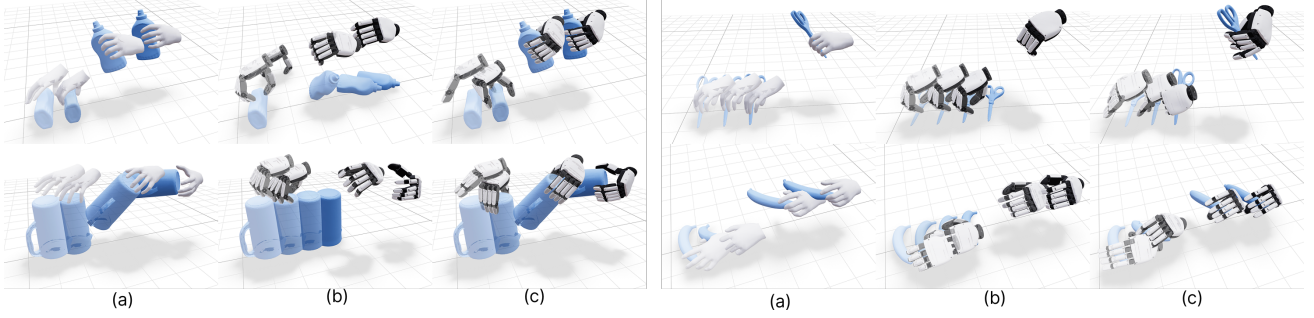


Figure 10: Execution of unseen DexYCB demonstrations. Transparent hands indicate motion over time. (a) Annotated human hand-object motion. (b) Open-loop kinematic replay fails to maintain contact in physics simulation. (c) Our reactive primitive acquires stable contact and executes the demonstrated object trajectory.

Table 4: Ablation results. Success rate (%) on three left hands.

Variant	XHand	Sharpa	Allegro	Avg.
Full GOTT	74.1 $\pm$ 1.2	75.0 $\pm$ 1.1	87.1 $\pm$ 4.4	78.7 $\pm$ 1.1
w/o link position obs.	49.4 $\pm$ 6.4	57.9 $\pm$ 10.3	75.7 $\pm$ 10.6	61.0 $\pm$ 4.4
proprio history length = 1	56.5 $\pm$ 5.1	65.0 $\pm$ 2.1	67.5 $\pm$ 3.0	63.0 $\pm$ 3.3
OHRA-style [54] obs.	24.9 $\pm$ 22.8	31.2 $\pm$ 25.2	20.9 $\pm$ 15.6	25.7 $\pm$ 18.3
w/o r_{contact}	0.0 $\pm$ 0.0	0.0 $\pm$ 0.0	0.0 $\pm$ 0.0	0.0 $\pm$ 0.0
w/o r_{lift}	0.8 $\pm$ 0.1	2.7 $\pm$ 1.3	1.0 $\pm$ 1.2	1.5 $\pm$ 0.8
w/o r_{dist}	70.6 $\pm$ 3.0	76.9 $\pm$ 4.2	80.6 $\pm$ 7.0	76.0 $\pm$ 1.6

with our policy, and then track the object trajectory. We use the same policy for all sequences without fine-tuning. We compare against kinematic replay, which directly executes the retargeted trajectories. Neither kinematic replay nor GOTT performs per-demonstration test-time optimization. We additionally report SPIDER [41], which generates robot controls via per-demonstration physics-informed optimization, as a reference with an additional test-time optimization stage. We use SPIDER’s task-success criterion and released MuJoCo implementation. Our reimplementations of SPIDER in IsaacLab and ManiSkill failed to achieve stable execution, so we omit their results. Our policy is trained in IsaacLab and evaluated zero-shot in ManiSkill and MuJoCo to assess sim-to-sim transfer.

On these 499 demonstrations, kinematic replay achieves a success rate of 12.6% in MuJoCo, whereas GOTT achieves 71.5% in IsaacLab, 72.9% in ManiSkill, and 69.7% in MuJoCo, demonstrating sim-to-sim transfer. SPIDER achieves 76.0% in MuJoCo but requires per-demonstration optimization and reoptimization whenever nominal object states or trajectories change. Figure 10 compares human demonstrations, kinematic replay, and executions by GOTT.

4.6. Ablations and Analysis

We analyze two aspects of GOTT: the primitive’s observation and reward design (Section 3.2) and the role of affordance reasoning and trajectory-aware selection in generating functional, feasible high-level specifications (Section 3.3).

Table 5: **Affordance- and Future-aware selection on Drill Use (20 scenes)**. Random and IK-only use affordance-agnostic candidates and select by random sampling and trajectory feasibility, respectively. VLM variants first generate candidates from a task-relevant contact region and then apply future-aware IK selection.

Selection	Affordance	Future-aware	gSR (%)	tjSR (%)	func. (%)	tSR (%)
Random	✗	✗	100	80	35	20
IK-only	✗	✓	95	90	30	30
VLM (gpt-5.6-sol)	✓	✓	90	70	80	55
VLM (gemini-flash-latest)	✓	✓	70	50	65	50
VLM (claude-opus-4-7)	✓	✓	80	70	85	65
VLM (claude-opus-5)	✓	✓	95	75	85	60

4.6.1. Contact primitive design

We vary one observation or reward setting at a time, holding all others fixed (Table 4). Removing link-position observations degrades success across hands. Reducing the proprioceptive history to one timestep removes temporal contact cues, lowering average success by 15.7 percentage points. We also compare with an OHRA-style [54] observation representation. In its original form, OHRA encodes only the object point cloud with a heavyweight point-cloud encoder, concatenates the resulting feature with a morphology VAE embedding, and uses the combined representation to predict grasps offline. This encoder is too large to train jointly with PPO at our training scale, so we adapt the representation to RL by (i) replacing the encoder with a lighter PointNet and (ii) adding the hand point cloud as input, while preserving OHRA’s morphology VAE conditioning. Even after this adaptation, average success drops to 25.7%, suggesting that per-link contact geometry provides a better representation for a closed-loop primitive.

Removing either r_{contact} or r_{lift} reduces average success to below 2%, confirming their necessity. While r_{dist} modestly improves lifting on training hands on average, it suppresses finger inversion and distorted joint bending (Figure 6). Under morphology shift, removing this regularization lowers average success on three unseen Allegro variants from 72.1% to 55.1%.

4.6.2. Affordance- and Future-Aware Grasp Selection

We evaluate affordance-conditioned candidate generation and future-aware selection on 20 matched *Drill Use* scenes, which require grasping the handle instead of the chuck. All methods use the same scene initializations, with position randomized by ± 0.10 m along x and y and orientation by $\pm 30^\circ$. Table 5 reports grasp success rate (gSR), task trajectory success rate (tjSR), functional correctness (func.), and overall task success rate (tSR). Trajectory success requires the drill to rotate by 45° about the specified axis while remaining grasped by the robot. Overall task success requires grasp success, trajectory success, and an affordance-correct grasp to all hold in the same trial. All rates are computed over the same 20 scenes. Scenes with no valid candidate terminate without execution and count as failures for all metrics.

Random selection achieves 100% gSR but only 20% tSR: an initially valid grasp may not support the task. Future-aware IK selection raises tjSR from 80% to 90%, but functional correctness and overall task success remain at 30%: successful trajectory execution alone does not ensure a task-appropriate grasp. Across VLM backbones, affordance conditioning improves func. to 65–85% and tSR to 50–65%. Restricting candidates to task-relevant regions lowers tjSR to 50–75% but improves overall task success.

5. Conclusion

We presented GOTT, a *reach-acquire-move* framework for dexterous manipulation from object trajectories and flexible reach specifications. At its core, a cross-embodiment contact-acquisition primitive establishes stable contact from approximate reach specifications, decoupling local contact control from arm-specific reaching and downstream tracking. Simulation and real-world experiments demonstrate the primitive’s reuse across objects, embodiments, tasks, workspaces, and high-level input sources.

Limitations and future work. First, high-level planning and contact acquisition operate sequentially. Tighter feedback between them could enable online recovery from poor reach specifications. Second, trajectory tracking relies on visual object pose estimation, which can fail under severe hand-object occlusion. Incorporating tactile sensing or visuotactile state estimation is a natural next step. Finally, we treat the hand-object transform as approximately fixed after grasping. Extending GOTT to handle changes in this transform would enable more precise in-hand manipulation tasks, such as unscrewing.

Acknowledgments

We thank Tingfan Wu for his help with hardware support. We are grateful to Shuo Chen for proposing to use a vision-language model to query task-relevant keypoints, as well as for insightful discussions and suggestions on VLM-based affordance reasoning. We thank Qiyang Yan for substantially accelerating the FoundationPose tracking speed, and Jiayuan Mao and Ruoshi Liu for helpful discussions and suggestions.

References

- [1] Jianibieke Adalibieke, Qianwei Han, Xueyi Liu, Yuzhe Qin, and Li Yi. AdaDexTrack: Dynamic modulation for adaptive and generalizable dexterous manipulation tracking. In *CVPR*, 2026.
- [2] Ananye Agarwal, Shagun Uppal, Kenneth Shaw, and Deepak Pathak. Dexterous functional grasping. In *CoRL*, 2023.
- [3] Prithviraj Banerjee, Sindi Shkodrani, Pierre Moulon, Shreyas Hampali, Fan Zhang, Jade Fountain, Edward Miller, Selen Basol, Richard Newcombe, Robert Wang, Jakob Julian Engel, and Tomas Hodan. Introducing hot3d: An egocentric dataset for 3d hand and object tracking. *arXiv:2406.09598*, 2024.
- [4] Homanga Bharadhwaj, Roozbeh Mottaghi, Abhinav Gupta, and Shubham Tulsiani. Track2act: Predicting point tracks from internet videos enables generalizable robot manipulation. In *ECCV*, 2024.
- [5] Berk Calli, Arjun Singh, James Bruce, Aaron Walsman, Kurt Konolige, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Yale-cmu-berkeley dataset for robotic manipulation research. *IJRR*, 2017.
- [6] Yu-Wei Chao, Wei Yang, Yu Xiang, Pavlo Molchanov, Ankur Handa, Jonathan Tremblay, Yashraj S Narang, Karl Van Wyk, Umar Iqbal, Stan Birchfield, Jan Kautz, and Dieter Fox. Dexycb: A benchmark for capturing hand grasping of objects. In *CVPR*, 2021.
- [7] Jiayi Chen, Yubin Ke, and He Wang. Bodex: Scalable and efficient robotic dexterous grasp synthesis using bilevel optimization. In *ICRA*, 2025.

- [8] Sirui Chen, Jeannette Bohg, and C Karen Liu. Springgrasp: Synthesizing compliant, dexterous grasps under shape uncertainty. In *RSS*, 2024.
- [9] Tao Chen, Megha Tippur, Siyang Wu, Vikash Kumar, Edward Adelson, and Pulkit Agrawal. Visual dexterity: In-hand reorientation of novel and complex object shapes. *Science Robotics*, 2023.
- [10] Yuanpei Chen, Chen Wang, Yaodong Yang, and C. Karen Liu. Object-centric dexterous manipulation from human motion data. In *CoRL*, 2024.
- [11] Sammy Christen, Muhammed Kocabas, Emre Aksan, Jemin Hwangbo, Jie Song, and Otmar Hilliges. D-Grasp: Physically plausible dynamic grasp synthesis for hand-object interactions. In *CVPR*, 2022.
- [12] Murtaza Dalal, Min Liu, Walter Talbott, Chen Chen, Deepak Pathak, Jian Zhang, and Ruslan Salakhutdinov. Local policies enable zero-shot long-horizon manipulation. In *ICRA*, 2025.
- [13] Vincent de Bakker, Joey Hejna, Tyler Lum, Onur Celik, Aleksandar Taranovic, Denis Blessing, Gerhard Neumann, Jeannette Bohg, and Dorsa Sadigh. Scaffolding dexterous manipulation with vision-language models. In *NeurIPS*, 2026.
- [14] Hao-Shu Fang, Hengxu Yan, Zhenyu Tang, Hongjie Fang, Chenxi Wang, and Cewu Lu. Anydexgrasp: General dexterous grasping for different hands with human-level learning efficiency. *arXiv:2502.16420*, 2025.
- [15] Yunhai Feng, Natalie Leung, Jiaxuan Wang, Lujie Yang, Haozhi Qi, and Preston Culbertson. A minimalist retargeting-guided reinforcement learning recipe for dexterous manipulation. In *CoRL*, 2026.
- [16] Google DeepMind. Gemini 3.1 pro model card. <https://deepmind.google/models/model-cards/gemini-3-1-pro/>, 2026.
- [17] Harsh Gupta, Mohammad Amin Mirzaee, and Wenzhen Yuan. Grasp to act: Dexterous grasping for tool use in dynamic settings. *RA-L*, 2026.
- [18] Harsh Gupta, Guanya Shi, and Wenzhen Yuan. LUCID: Learning embodiment-agnostic intent models from unstructured human videos for scalable dexterous robot skill acquisition. In *CoRL*, 2026.
- [19] Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, Yashraj Narang, Jean-Francois Lafleche, Dieter Fox, and Gavriel State. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. In *ICRA*, 2023.
- [20] Nick Heppert, Max Argus, Tim Welschhold, Thomas Brox, and Abhinav Valada. Ditto: Demonstration imitation by trajectory transformation. In *IROS*, 2024.
- [21] Cheng-Chun Hsu, Bowen Wen, Jie Xu, Yashraj Narang, Xiaolong Wang, Yuke Zhu, Joydeep Biswas, and Stan Birchfield. SPOT: SE(3) pose trajectory diffusion for object-centric manipulation. In *ICRA*, 2025.
- [22] Linyi Huang, Hui Zhang, Zijian Wu, Sammy Christen, and Jie Song. FunGrasp: Functional grasping for diverse dexterous hands. *RA-L*, 2025.

- [23] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. In *CoRL*, 2023.
- [24] Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. In *CoRL*, 2024.
- [25] Bingxin Ke, Dominik Narnhofer, Shengyu Huang, Lei Ke, Torben Peters, Katerina Fragkiadaki, Anton Obukhov, and Konrad Schindler. Video depth without video models. In *CVPR*, 2025.
- [26] Kushal Kedia, Tyler Ga Wei Lum, Jeannette Bohg, and C Karen Liu. Simtoolreal: An object-centric policy for zero-shot dexterous tool manipulation. In *RSS*, 2026.
- [27] Justin Kerr, Chung Min Kim, Mingxuan Wu, Brent Yi, Qianqian Wang, Ken Goldberg, and Angjoo Kanazawa. Robot see robot do: Imitating articulated object manipulation with monocular 4d reconstruction. In *CoRL*, 2024.
- [28] Yuxuan Kuang, Sungjae Park, Katerina Fragkiadaki, and Shubham Tulsiani. Dex4d: Task-agnostic point track policy for sim-to-real dexterous manipulation. In *RSS*, 2026.
- [29] 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 *CoRL*, 2024.
- [30] Kailin Li, Puhao Li, Tengyu Liu, Yuyang Li, and Siyuan Huang. Maniptrans: Efficient dexterous bimanual manipulation transfer via residual learning. In *CVPR*, 2025.
- [31] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *ICRA*, 2023.
- [32] Xueyi Liu, Jianibieke Adalibieke, Qianwei Han, Yuzhe Qin, and Li Yi. Dextrack: Towards generalizable neural tracking control for dexterous manipulation from human references. In *ICLR*, 2025.
- [33] Tyler Ga Wei Lum, Albert H Li, Preston Culbertson, Krishnan Srinivasan, Aaron D Ames, Mac Schwager, and Jeannette Bohg. Get a grip: Multi-finger grasp evaluation at scale enables robust sim-to-real transfer. In *CoRL*, 2024.
- [34] Tyler Ga Wei Lum, Martin Matak, Viktor Makoviychuk, Ankur Handa, Arthur Allshire, Tucker Hermans, Nathan D Ratliff, and Karl Van Wyk. Dextrah-g: Pixels-to-action dexterous arm-hand grasping with geometric fabrics. In *CoRL*, 2024.
- [35] 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 *CoRL*, 2025.
- [36] Zhengyi Luo, Jinkun Cao, Sammy Joe Christen, Alexander Winkler, Kris Kitani, and Weipeng Xu. Omnigrasp: Grasping diverse objects with simulated humanoids. In *NeurIPS*, 2024.
- [37] Zhao Mandi, Yifan Hou, Dieter Fox, Yashraj Narang, Ajay Mandlekar, and Shuran Song. Dexmachina: Functional retargeting for bimanual dexterous manipulation. *arXiv:2505.24853*, 2025.

- [38] Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Muñoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, Lukasz Wawrzyniak, Milad Rakhsha, Alain Denzler, Eric Heiden, Ales Borovicka, Ossama Ahmed, Ireteayo Akinola, Abrar Anwar, Mark T. Carlson, Ji Yuan Feng, Animesh Garg, Renato Gasoto, Lionel Gulich, Yijie Guo, M. Gussert, Alex Hansen, Mihir Kulkarni, Chenran Li, Wei Liu, Viktor Makoviychuk, Grzegorz Malczyk, Hammad Mazhar, Masoud Moghani, Adithyavairavan Murali, Michael Noseworthy, Alexander Poddubny, Nathan Ratliff, Welf Rehberg, Clemens Schwarke, Ritvik Singh, James Latham Smith, Bingjie Tang, Ruchik Thaker, Matthew Trepte, Karl Van Wyk, Fangzhou Yu, Alex Millane, Vikram Ramasamy, Remo Steiner, Sangeeta Subramanian, Clemens Volk, CY Chen, Neel Jawale, Ashwin Varghese Kuruttukulam, Michael A. Lin, Ajay Mandlekar, Karsten Patzwaldt, John Welsh, Huihua Zhao, Fatima Anes, Jean-Francois Lafleche, Nicolas Moënné-Loccoz, Soowan Park, Rob Stepinski, Dirk Van Gelder, Chris Amever, Jan Carius, Jumyung Chang, Anka He Chen, Pablo de Heras Ciechowski, Gilles Daviet, Mohammad Mohajerani, Julia von Muralt, Viktor Reutskyy, Michael Sauter, Simon Schirm, Eric L. Shi, Pierre Terdiman, Kenny Vilella, Tobias Widmer, Gordon Yeoman, Tiffany Chen, Sergey Grizan, Cathy Li, Lotus Li, Connor Smith, Rafael Wiltz, Kostas Alexis, Yan Chang, David Chu, Linxi Jim Fan, Farbod Farshidian, Ankur Handa, Spencer Huang, Marco Hutter, Yashraj Narang, Soha Pouya, Shiwei Sheng, Yuke Zhu, Miles Macklin, Adam Moravanszky, Philipp Reist, Yunrong Guo, David Hoeller, and Gavriel State. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv:2511.04831*, 2025.
- [39] OpenAI. Gpt-5.5 system card. <https://openai.com/index/gpt-5-5-system-card/>, 2026.
- [40] OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafał Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *IJRR*, 2020.
- [41] 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. In *IROS*, 2026.
- [42] Austin Patel and Shuran Song. GET-Zero: Graph embodiment transformer for zero-shot embodiment generalization. In *ICRA*, 2025.
- [43] Shivansh Patel, Shraddhaa Mohan, Hanlin Mai, Unnat Jain, Svetlana Lazebnik, and Yunzhu Li. Robotic manipulation by imitating generated videos without physical demonstrations. In *ICLR*, 2026.
- [44] Haozhi Qi, Ashish Kumar, Roberto Calandra, Yi Ma, and Jitendra Malik. In-hand object rotation via rapid motor adaptation. In *CoRL*, 2022.
- [45] Haozhi Qi, Brent Yi, Sudharshan Suresh, Mike Lambeta, Yi Ma, Roberto Calandra, and Jitendra Malik. General in-hand object rotation with vision and touch. In *CoRL*, 2023.
- [46] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. In *ICLR*, 2025.

- [47] Javier Romero, Dimitrios Tzionas, and Michael J Black. Embodied hands: Modeling and capturing hands and bodies together. *arXiv preprint arXiv:2201.02610*, 2022.
- [48] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.
- [49] Balakumar Sundaralingam, Siva Kumar Sastry Hari, Adam Fishman, Caelan Garrett, Karl Van Wyk, Valts Blukis, Alexander Millane, Helen Oleynikova, Ankur Handa, Fabio Ramos, Nathan Ratliff, and Dieter Fox. Curobo: Parallelized collision-free robot motion generation. In *ICRA*, 2023.
- [50] Weikang Wan, Haoran Geng, Yun Liu, Zikang Shan, Yaodong Yang, Li Yi, and He Wang. UniDex-Grasp++: Improving dexterous grasping policy learning via geometry-aware curriculum and iterative generalist-specialist learning. In *ICCV*, 2023.
- [51] 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 *ICRA*, 2023.
- [52] Yi-Lin Wei, Mu Lin, Yuhao Lin, Jian-Jian Jiang, Xiao-Ming Wu, Ling-An Zeng, and Wei-Shi Zheng. AffordDexGrasp: Open-set language-guided dexterous grasp with generalizable-instructive affordance. In *ICCV*, 2025.
- [53] Zhenyu Wei, Zhixuan Xu, Jingxiang Guo, Yiwen Hou, Chongkai Gao, Zhehao Cai, Jiayu Luo, and Lin Shao. $\mathcal{D}(\mathcal{R}, \mathcal{O})$ grasp: A unified representation of robot and object interaction for cross-embodiment dexterous grasping. In *ICRA*, 2025.
- [54] Zhenyu Wei, Yunchao Yao, and Mingyu Ding. One hand to rule them all: Canonical representations for unified dexterous manipulation. In *RSS*, 2026.
- [55] Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects. In *CVPR*, 2024.
- [56] Chuan Wen, Xingyu Lin, John So, Kai Chen, Qi Dou, Yang Gao, and Pieter Abbeel. Any-point trajectory modeling for policy learning. In *RSS*, 2024.
- [57] Zhiyuan Wu, Xiangyu Zhang, Zhuo Chen, Jiankang Deng, Rolandos Alexandros Potamias, and Shan Luo. Unimorphgrasp: Diffusion model with morphology-awareness for cross-embodiment dexterous grasp generation. *arXiv preprint arXiv:2602.00915*, 2026.
- [58] Mengda Xu, Zhenjia Xu, Yinghao Xu, Cheng Chi, Gordon Wetzstein, Manuela Veloso, and Shuran Song. Flow as the cross-domain manipulation interface. In *CoRL*, 2024.
- [59] Yinzhen Xu, Weikang Wan, Jialiang Zhang, Haoran Liu, Zikang Shan, Hao Shen, Ruicheng Wang, Haoran Geng, Yijia Weng, Jiayi Chen, Tengyu Liu, Li Yi, and He Wang. UniDexGrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy. In *CVPR*, 2023.
- [60] Lixin Yang, Kailin Li, Xinyu Zhan, Fei Wu, Anran Xu, Liu Liu, and Cewu Lu. Oakink: A large-scale knowledge repository for understanding hand-object interaction. In *CVPR*, 2022.

- [61] Haoqi Yuan, Bohan Zhou, Yuhui Fu, and Zongqing Lu. Cross-embodiment dexterous grasping with reinforcement learning. In *ICLR*, 2025.
- [62] Hui Zhang, Sammy Christen, Zicong Fan, Otmar Hilliges, and Jie Song. GraspXL: Generating grasping motions for diverse objects at scale. In *ECCV*, 2024.
- [63] Hui Zhang, Zijian Wu, Linyi Huang, Sammy Christen, and Jie Song. Robustdexgrasp: Robust dexterous grasping of general objects. In *CoRL*, 2025.
- [64] Hui Zhang, Mirko Meboldt, and Jie Song. ConGraspXL: Controllable constraint-conditioned dexterous grasping motion synthesis. *arXiv:2609.16319*, 2026.
- [65] Yifeng Zhu, Arisrei Lim, Peter Stone, and Yuke Zhu. Vision-based manipulation from single human video with open-world object graphs. *Autonomous Robots*, 2026.

A. Reinforcement Learning Details

A.1. Reward Function Details

The per-step reward is

$$r = w_d r_{\text{dist}} + w_c r_{\text{contact}} + w_l r_{\text{lift}} + w_o^p r_{\text{obj}}^p + w_o^r r_{\text{obj}}^r, \quad (1)$$

Reward weights are listed in Table 6.

Distance reward r_{dist} . A negative weighted sum of link-to-surface distances, each clipped at 0.1 m:

$$r_{\text{dist}} = - \sum_l \omega_l \min(\|\mathbf{a}_l\|, 0.1),$$

with unnormalized per-link weights $\omega_{\text{thumb-tip}} = 8$, $\omega_{\text{other fingertips}} = 4$, $\omega_{\text{palm}} = 0$, and $\omega_l = 1$ for all remaining links. In cross-embodiment training, the weights are masked to active links and normalized to sum to 1 per environment. For each canonical link l at world position $\mathbf{p}_l$, $\mathbf{a}_l$ is the displacement to the nearest of $M=512$ sampled object-surface points, and $\|\mathbf{a}_l\|$ is its Euclidean norm. The point cloud is stored in the object frame and transformed to the world frame using the tracked object pose.

Contact reward r_{contact} . Encourages both thumb contact and an opposing grasp, in which the thumb and at least one non-thumb finger simultaneously contact the object:

$$r_{\text{contact}} = \mathbf{1}[\text{thumb in contact}] + \mathbf{1}[\text{is_grasping}],$$

where a finger is considered to be in contact if its contact force with the object exceeds $\epsilon_F = 0.05$ N, and $\text{is_grasping} = (\text{thumb in contact}) \wedge (\exists \text{ non-thumb finger in contact})$.

Lifting reward r_{lift} . A grasp-gated, scaled reward proportional to the object’s height above its post-reset position:

$$r_{\text{lift}} = \mathbf{1}[\text{is_grasping}] \cdot \text{clip}(100 \Delta h, 0, 100 h_{\text{thr}}), \quad \Delta h = \max(0, z - z_0),$$

where z is the object’s current height, z_0 is its height immediately after reset, and $h_{\text{thr}} = 0.05$ m (so the clip ceiling is 5.0). The indicator $\mathbf{1}[\text{is_grasping}]$ ensures that lifting is rewarded only when the object is grasped. Clipping prevents additional reward from throwing the object upward.

Object stability reward r_{obj} . Exponential “stay-put” bonuses on planar displacement and rotation from the post-reset pose:

$$r_{\text{obj}}^p = e^{-10 \|\Delta \mathbf{xy}\|}, \quad r_{\text{obj}}^r = e^{-\Delta \theta(\mathbf{q}, \mathbf{q}_0)},$$

where $\Delta \mathbf{xy} \in \mathbb{R}^2$ is the planar displacement from the object’s post-reset position, and $\Delta \theta(\mathbf{q}, \mathbf{q}_0) = 2 \arccos(|\langle \mathbf{q}, \mathbf{q}_0 \rangle|)$ is the geodesic angle between its current and post-reset orientations, represented by unit quaternions. These rewards discourage sliding or rotating the object during contact acquisition, which we found important for stable real-world execution.

Curriculum / annealing. The object-stability weights are linearly annealed from an initial value to the final value over T_{cur} per-environment control steps: $w(t) = w_{\text{init}}(1 - \rho) + w_{\text{final}}\rho$, $\rho = \min(1, t/T_{\text{cur}})$. We anneal w_o^p, w_o^r from $w_{\text{init}} = 2.0$ to $w_{\text{final}} = 10.0$ over $T_{\text{cur}} = 10^5$ steps. The distance, contact, and lifting weights are not annealed.

Table 6: Reward weights.

Symbol	Term	Value
w_d	distance	16.0
w_c	contact	4.0
w_o^p	object position stability	10.0 (init. 2.0)
w_o^r	object rotation stability	10.0 (init. 2.0)
w_l	lifting	1.0
T_{cur}	annealing steps	10^5

A.2. Cross-Embodiment Unified Policy Representation

Canonical joint space. We map every hand onto a fixed *canonical* joint layout consisting of $J_{\text{can}} = 26$ slots partitioned into five contiguous per-finger blocks (6 for the thumb and 5 for each other finger). The wrist is excluded from this layout and controlled separately through a shared 6-DoF floating-base interface. Each hand fills only the slots corresponding to its native joints, and the remaining slots are masked. The number of active slots equals the hand’s DoF: XHand 12, Allegro 16, and Sharpa Wave 22 (Table 7).

Table 7: **Canonical 26-DoF joint layout and per-hand instantiation.** Each finger is allocated a fixed number of slots in the canonical layout (6 for the thumb, 5 for each non-thumb finger); slots are filled by a hand’s native joint when present, and masked ([]) otherwise. F, A, and R denote flexion, abduction/adduction, and axial rotation respectively. Reading order within each finger is base $\rightarrow$ tip.

Hand	DoF	Thumb (6)	Index (5)	Middle (5)	Ring (5)	Pinky (5)
Allegro	16	RAF[]F[]	R[]FFF	R[]FFF	R[]FFF	[] [] [] [] []
Sharpa Wave	22	[] AF A F F	[] A F F F	[] A F F F	[] A F F F	F A F F F
XHand	12	[] AF [] F []	[] A F F []	[] [] F F []	[] [] F F []	[] [] F F []

Canonical link space. For the canonical link ordering, we map each robot’s native link positions (obtained via forward kinematics from its URDF) onto a fixed $L_{\text{can}} = 21$ keypoint skeleton following the standard hand topology used in MANO [47]: one palm keypoint (canonical index 0) plus four keypoints per finger (5×4) ordered base-to-tip, with fingertips at canonical indices $\{4, 8, 12, 16, 20\}$. Link observations and link-based rewards use this canonical ordering.

Per-robot canonical mapping. Each robot r stores an index map π_r and a sign vector s_r with entries in $\{-1, +1\}$ that translate between its native joints and the canonical slots. The policy reads and writes in the *canonical* space. At actuation, the canonical hand command $a^{\text{hand}} \in \mathbb{R}^{J_{\text{can}}}$ is gathered into native joint order and sign-corrected to account for differences in native joint-axis conventions:

$$a_r^{\text{native}} = a^{\text{hand}}[\pi_r] \odot s_r.$$

Masks and padding. For each robot we form binary masks $\delta^{\text{joint}} \in \{0, 1\}^{26}$ and $\delta^{\text{link}} \in \{0, 1\}^{21}$ that flag occupied canonical slots. Both masks are appended to the observation so that the policy can identify active dimensions. Unoccupied joint slots are filled with zeros, and missing link positions are filled using the nearest existing link.

A.3. Observations

The observation concatenates three blocks: a 5-frame proprioceptive history, single-frame hand-object displacement vectors, and joint and link masks. The policy does not receive the object pose explicitly. Object information enters only through the per-link nearest-surface displacement vectors $\mathbf{a}_l$ defined in the distance reward above. Within the history buffer, frames are ordered from oldest to newest, $[o_{t-K+1}, \dots, o_t]$ with history length $K = 5$. At episode start, the buffer is padded with copies of the first observed frame. The full observation breakdown is given in Table 8.

Table 8: Observation components. $K = 5$.

Component	Composition	Dim
Joint positions	26 canonical $\times K$ frames	$26K$
Joint targets	26 canonical $\times K$ frames	$26K$
Link positions	21 links $\times 3 \times K$ frames	$63K$
Hand-object vectors	21 links $\times 1$ nearest pt $\times 3$	63
Joint mask	active canonical joints	26
Link mask	active canonical links	21

Joint positions and joint targets. Joint positions and previous-step joint targets are reordered into the canonical 26-joint layout and padded with zeros for joints not present on the active hand.

Link positions. We use forward kinematics to compute the 3D world positions of links in the canonical 21-keypoint topology, discarding their orientations. Missing keypoints are padded using the nearest existing link, as described in Section A.2.

Hand-object vectors (following [63]). For each of the 21 canonical links we compute the vector to its *nearest* point on the object surface point cloud ($N = 512$ points sampled per object), giving a 21×3 tensor.

Normalization. There is no in-environment normalization. `r1_games` normalizes observations and value targets using running means and standard deviations. Observations are clipped to $[-5, 5]$.

A.4. Policy Action

Action space. The action is $a = [a^{\text{EE}}(6), a^{\text{hand}}(J_{\text{can}})]$, i.e. $6 + 26 = 32$ dimensions. The first 6 dimensions specify an end-effector *delta* (3 translation components and 3 Euler angles) for the shared floating base, converted to the 6 virtual base-joint targets. The hand block is in canonical space and remapped per robot as in Section A.2.

Range, scaling, and clipping. Raw policy actions are clipped to $[-1, 1]$ before being rescaled into physical commands. All controllers operate in residual (delta-target) mode: each scaled delta is added to the previous control target, which is then clamped to the corresponding physical limit. For the hand, per-joint deltas are rescaled using each joint’s upper and lower limits so that an action of ± 1 corresponds to a full-range increment. The end-effector deltas are scaled to ± 0.008 m per step for translation and ± 0.052 rad per step for rotation, then composed with the previous target pose.

Control frequency. Simulation uses a timestep of $dt = 1/120$ s. With `decimation = 12`, each action is held for 12 physics substeps, giving a control rate of 10 Hz.

A.5. Reset and Initialization

Length and termination. We use a fixed episode horizon of $H = 60$ control steps (corresponding to 6.0 s at 10 Hz). An episode terminates when the step counter reaches H . At the start of training, episode step counters across the 4,096 parallel environments are initialized uniformly at random in $\{0, 1, \dots, H-1\}$ rather than synchronously at zero. This staggers episode resets so that rollout batches contain transitions from different phases (approach, contact, and lifting), improving sample diversity and stabilizing early training.

Reset distribution. At scene initialization, each environment randomly selects a robot embodiment and an object. At each reset, it samples a valid pair (object resting pose, initial hand configuration) from the per-object grasp dataset. We further augment the initial state with a random planar translation $\Delta \mathbf{p}_{xy} \sim \mathcal{U}([-0.05, 0.05]^2)$ m and a random yaw $\phi \sim \mathcal{U}(-\pi, \pi)$ about the world z -axis. The same transformation is applied to both components, preserving their relative geometry.

A.6. Domain Randomization

We apply domain randomization to encourage robustness and sim-to-real transfer. All physics parameters are resampled at every environment reset. Friction and restitution are set directly to values sampled from the listed ranges. Mass, scale, and actuator gains are scaled relative to their nominal values for each robot or object. The nominal object density is $\rho = 200 \text{ kg/m}^3$, which determines each object’s baseline mass before the $\times [0.5, 3.0]$ mass scale is applied.

Table 9: **Domain randomization ranges.** All distributions are uniform. Multiplicative ranges ($\times$) scale the per-object or per-robot nominal value; absolute ranges directly set the parameter value.

Group	Parameter	Range	Type
Object	Scale	[0.96, 1.04]	$\times$ nominal
	Mass	[0.5, 3.0]	$\times$ nominal (from $\rho \cdot V \cdot \text{scale}^3$)
	Static/dynamic friction	[0.5, 1.5]	absolute
	Restitution	[0.0, 0.1]	absolute
Robot links	Static/dynamic friction	[0.5, 2.0]	absolute
	Restitution	[0.1, 0.5]	absolute
Actuators	Stiffness (k_p)	[0.5, 2.0]	$\times$ nominal
	Damping (k_d)	[0.3, 3.0]	$\times$ nominal
Perturbation	Push force (max)	2.0 N	sampled uniformly per axis
	Application offset	± 0.05 m	from object center

Nominal actuator gains. The floating arm base has $k_p = [1000, 1000, 1000, 300, 300, 300]$ and $k_d = [100, 100, 100, 15, 15, 15]$ for its six virtual joints. The per-hand k_p and k_d values are obtained through system identification on the physical hardware.

A.7. Contact Prediction Details

We train a separate contact predictor using rollouts from the frozen policy after convergence. The predictor is a 2-layer MLP with a hidden size of 256 and ELU activation. It takes joint positions, joint

targets, and link positions over the most recent $k = 5$ steps and outputs five per-finger contact logits. We train it with binary cross-entropy. The ground-truth label for each finger is 1 if the contact-sensor force exceeds $\epsilon_F = 0.05$ N.

Training. At each epoch, we collect 120 steps using deterministic actions from the frozen policy across 4,096 parallel environments. We train the predictor on the resulting in-memory buffer in shuffled mini-batches of 4,096 with Adam at a learning rate of 3×10^{-4} . We train for 200 epochs in total.

Deployment. At inference, contact probabilities are obtained by sigmoid activation and thresholded at 0.5. We set `is_grasping` to true when the thumb and at least one non-thumb finger are predicted to be in contact. We switch to tracking when `is_grasping` remains true for five consecutive control steps.

A.8. Additional Training Details

Network architecture. We use a feed-forward MLP with hidden sizes [512, 512, 256, 128]. Training hyperparameters are listed in Table 10.

Table 10: PPO hyperparameters.

Parameter	Value
Parallel environments	4096
Horizon (rollout) length	60
Minibatch size	8192
Mini-epochs per update	5
Discount γ	0.95
GAE λ (<code>tau</code>)	0.90
Learning rate	3×10^{-4} , adaptive (KL)
KL threshold	0.016
PPO clip ϵ	0.2
Value loss coef. (<code>critic_coef</code>)	4
Value clipping	enabled
Entropy coef.	0.0
Bounds-loss coef.	1×10^{-4}
Gradient norm clip	1.0
Normalize input / value / advantage	yes / yes / yes
Mixed precision	no
Max epochs	2000
Seed	42, 84, 126

Object set. We use 16 YCB objects, as illustrated in Figure 11. Each object has 2–7 distinct stable resting poses on the table (~ 100 object-pose configurations per hand).

Grasp data generation and filtering. For each (object, resting-pose) pair, we generate grasp candidates with BODex [7]. BODex uses a heuristic seed sampler to propose an initial hand configuration, which we keep for the reset distribution (Section A.5). The full bilevel grasp optimization then refines this seed into a converged *reference* grasp. We validate the reference grasps with BODex’s official evaluation code and retain 4,000 successful (initial configuration, reference grasp) pairs per (object, resting-pose) pair.



Figure 11: Simulation training assets.

Hardware and training duration. We train each policy in IsaacLab on a single NVIDIA L40S GPU for 2000 epochs, taking 24 hours. This corresponds to approximately 491.5 million environment steps, with 60 steps across 4096 parallel environments per epoch.

B. Future-Aware Planning Details

At deployment, the planner identifies a task-relevant contact region and then generates candidate initializations within this region. A vision-language model (VLM) provides the task-aware affordance selection, while candidate generation and trajectory selection are performed geometrically and kinematically.

Task-aware region selection. We render the object at its observed pose on the table from four side views (Figure 12 (a)). The views use textured rendering with a low camera elevation and are overlaid with a light pixel grid for image-coordinate reference. The VLM is prompted to identify the object’s functional parts, mark each visible part in each view, and designate the single part to be grasped for the specified task (Figure 12 (b)). For the designated part, the VLM additionally predicts multiple points spanning its extent. All predicted points are back-projected onto the object mesh by ray casting. For each functional part, its annotations across views are clustered in the object frame, and the mean of the largest cluster is used as the part location. Predicted grasp points whose back-projected locations are inconsistent with the designated part are discarded. We define the task-relevant contact region as the union of 3 cm surface neighborhoods around the remaining grasp points (Figure 12 (c)).

Candidate generation. Following the initialization stage of BODex [7], we sample candidate hand poses and joint configurations from surface points within the selected contact region. We include the table in the collision world and reject candidate initializations that collide with the table or other scene geometry, as well as ill-posed poses. We retain $N = 10$ valid candidate initializations by default. These candidates serve as approximate initial configurations for the contact-acquisition primitive rather than final grasps.

Trajectory expansion and selection. For each remaining candidate, we plan a collision-free approach from the robot’s initial configuration to the initial grasp pose with cuRobo [49]. We then assume a rigid hand-object attachment and compute the hand trajectory along the desired object motion τ_o , solving IK along this trajectory to recover the full arm-hand motion. Candidates whose approach or tracking

segment is infeasible are discarded. Among the remaining candidates, we select the trajectory with the smallest arm motion,

$$c_i^{\text{ik}} = \sum_{t=1}^T \|q_{i,t}^{\text{arm}} - q_{i,t-1}^{\text{arm}}\|_2,$$

where $q_{i,t}^{\text{arm}}$ denotes the arm configuration obtained by IK for candidate i at timestep t .

Prompt and example. We use Claude Opus 4.7 or Claude Opus 5 via the Anthropic API. The prompt specifies the manipulation task and provides generic guidance distinguishing the holding region from the working end of an object. Given the four rendered views, the model returns a JSON list of labeled object parts, with one pixel coordinate for each visible part in each view and a binary grasp flag indicating the part selected for grasping. Figure 12 shows an example from a Drill Use trial. The VLM identifies the pistol grip as the task-relevant holding region, and the subsequent candidate generation is restricted to this region.

You are labeling the functional KEYPOINTS of an object for a dexterous robot, then choosing which one to grasp.

Task: {task_name}

The object may have a designated HOLDING REGION (shaped to fit a hand) vs a WORKING END (does the task). Grasp the HOLDING REGION. [...]

You are given 4 rendered views of the SAME object standing on a table, from different angles. Each image is 640x480 px with a light coordinate grid [...]

Step 1 -- Identify the object's distinct FUNCTIONAL KEYPOINTS (the named parts a person would talk about). Give each a short label.

Step 2 -- For EACH keypoint, mark its pixel in EVERY view where it is clearly visible. All picks for one keypoint must be the SAME physical point, so we can triangulate it in 3D.

Step 3 -- Set "grasp": true on the ONE keypoint the hand should converge on to hold the object for this task, and false on the rest.

Step 4 -- For that grasp keypoint ONLY, also give "extent": 2-4 extra pixel picks that span the full length of the holding part (e.g. both ends and the middle of a grip or handle). Together with the keypoint they define WHERE the fingers may land, so cover the whole holdable part but do NOT extend into the working end or onto other parts.

Respond with ONLY a JSON object: {"keypoints": [{"label": <str>, "grasp": <bool>, "picks": [{"view": <int>, "u": <int>, "v": <int>}, ...], "extent": [{"view": <int>, "u": <int>, "v": <int>}, ...], "reason": "<why>"}] ("extent" only on the grasp keypoint; omit it elsewhere.)

C. Real-World Grasping Experiment Details

This section provides per-object real-world grasping results across the three arm-hand platforms (Table 11). Tianji-XHand (48/50, 96%) achieves higher overall success than both Franka platforms (45/50 each, 90%). We attribute this to the larger reachable workspace of the Tianji arm relative to our Franka setup.

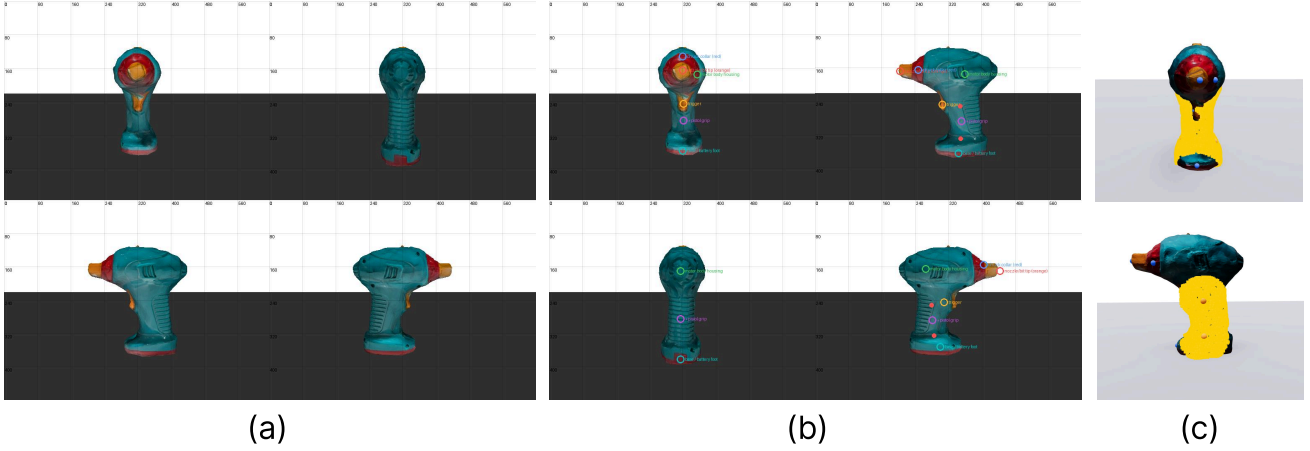


Figure 12: Task-aware region selection for Drill Use. (a) Four rendered views provided to the VLM. (b) The VLM identifies functional parts and selects the pistol grip as the task-relevant holding region, with multiple grasp points spanning the grip. (c) Back-projected grasp points define the contact region (yellow), from which candidate initializations are sampled.

Table 11: Per-object grasp success across embodiments (out of 5 trials). Objects are abbreviated as O1-O10: O1 mustard-bottle, O2 bleach-cleanser, O3 drill-red, O4 tomato-soup, O5 meat-can, O6 red-bottle, O7 drill-green, O8 apple-tea, O9 eraser, O10 maize. Objects O1–O5 are seen during training; O6–O10 are unseen.

Embodiment	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10	Total (%)
Franka-Sharpa	4	4	5	5	5	5	4	3	5	5	90
Franka-XHand	5	5	5	4	5	3	5	4	5	4	90
Tianji-XHand	5	5	4	4	5	5	5	5	5	5	96

D. Real-World Full-Pipeline Experiment Visualizations

We present qualitative comparisons of the full pipeline against the DRO and BODex baselines in Figure 13, illustrating three recurring failure modes: (i) grasping at an unintended contact region that does not support the downstream trajectory (Figure 13(d)), (ii) displacing the object during approach (Figure 13(a, b)), and (iii) forming a cage-like grasp without firm opposing finger-thumb contact (Figure 13(c)). In contrast, GOTT addresses these failure modes by (i) adjusting the wrist and fingers locally to recover from imperfect pre-grasp poses produced by upstream planning, (ii) reacting to object displacement during approach, and (iii) establishing firm opposing finger-thumb contact rather than caging the object loosely. We refer the reader to the supplementary video or our project website for video demonstrations of real-world performance.

E. Simulation Grasping Evaluation Protocol and Baseline Details

The grasp success rates we report for BODex [7] and DRO [53] in Table 1 of the main paper are lower than those reported in their original publications. This section clarifies the source of this gap and presents a controlled reproduction.

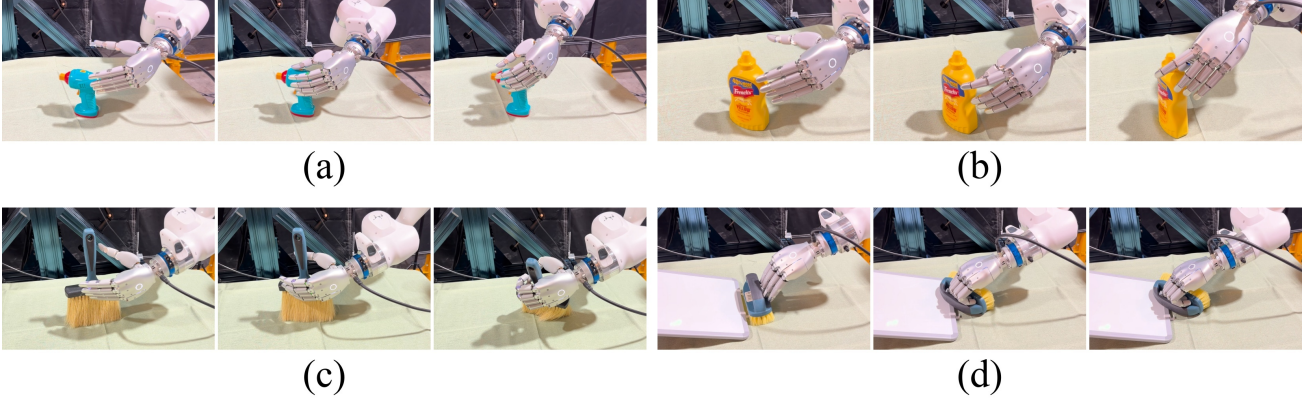


Figure 13: Qualitative comparison on real-world tasks. Subfigures (a)-(b) are from BODex, and (c)-(d) are from DRO.

E.1. Protocol Comparison with BODex

BODex’s original evaluation places each scaled object in a *floating state without a table*. Ten grasps are synthesized per (object, scale, pose) instance. The hand is initialized at the pre-grasp pose $\mathbf{x}_p$ and driven to the squeezed pose $\mathbf{x}_s$. Gravity is then applied along the positive and negative directions of each of the three coordinate axes. A grasp counts as successful only if the object’s translation and rotation deviations remain within 5 cm and 15° for at least 3 s under *all six* directions. In contrast, our protocol places objects on a tabletop under gravity from the start of the trial. Both seed-pose initialization (used identically across all methods) and grasp execution must avoid hand-table collisions throughout the trial. This setting is closer to real-world deployment, where objects rest on a workspace surface and the robot must operate within the resulting constraints. In summary, the two protocols differ in two respects:

- **Generation phase.** BODex synthesizes grasps without considering a table. Our setup adds hand-table collision constraints to both initial-pose sampling and the bilevel optimization.
- **Evaluation phase.** BODex evaluates grasps in a floating-hand state with gravity applied after squeezing along all six directions. Our setup places the object on a table under $-z$ gravity from the start of the trial.

E.2. Reproducing BODex under Its Evaluation Protocol

To isolate the effect of the generation procedure, we use BODex’s officially released code with its default parameters (in the main branch), synthesize 20 initial poses per object under two conditions, and evaluate both under BODex’s original protocol:

- **Floating generation** (BODex default): initial-pose sampling and bilevel optimization ignore the table.
- **Tabletop generation** (ours): initial-pose sampling and bilevel optimization avoid hand-table collisions.

Table 12 shows that tabletop-aware generation alone, with evaluation held fixed, drops the success rate from 70.7% to 57.1%. This degradation is attributable purely to the harder generation problem: the init-pose distribution changes and the bilevel optimization becomes more constrained. The floating-generation result is below the 79.9% reported in [7] because the released code’s main branch differs from the paper’s configuration (preserved in the baseline branch) in object mass (100 g vs. 30 g) and joint stiffness ($k_p=5$ vs. $k_p=2$); rerunning floating evaluation with the paper’s configuration (object mass

Table 12: **Effect of generation-time table constraints on BODex grasp success rate (%)**. Allegro right hand, 4 object scales, evaluated under BODex’s original protocol. Only the generation phase differs between the two columns. Objects are taken from DexGraspNet [51], normalised so that each object’s bounding-box diagonal measures 1 m, and then rescaled to four target diagonals (@006=0.06 m, @008=0.08 m, @010=0.10 m, @012=0.12 m).

Scale	@006	@008	@010	@012	avg.
Floating	69.79	73.79	72.09	67.26	70.73
Tabletop	53.29	60.90	60.05	54.28	57.13

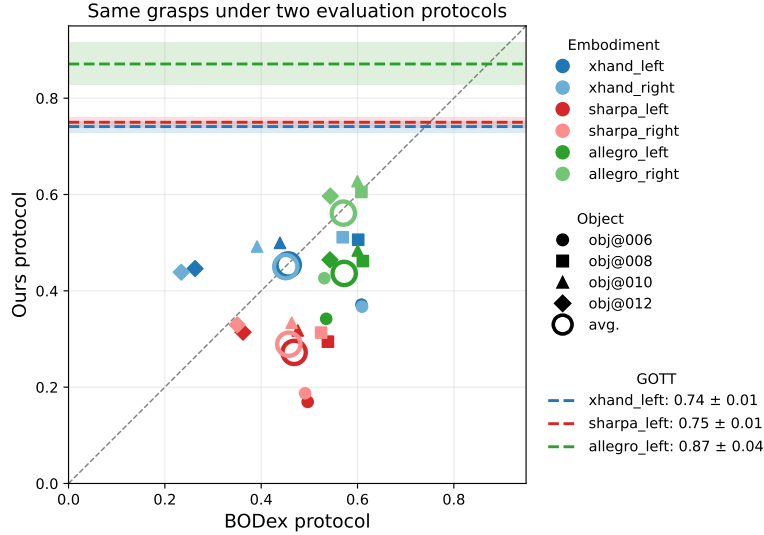


Figure 14: Evaluation protocol comparison. The same BODex grasps evaluated under BODex’s original protocol and our tabletop protocol.

30 g and $k_p=2$) yields 77.2%, with the residual gap likely due to seed variance. Following the authors’ recommendation, Table 12 and Figure 14 use the main-branch defaults; this choice does not affect the conclusions above, since both rows of Table 12 share the same configuration.

E.3. Evaluation-Phase Comparison

We further evaluate the same set of BODex-generated grasps under both protocols. Figure 14 plots, for each (embodiment, object set) pair, BODex’s success rate under its original protocol (x-axis) against the success rate of the *same grasps* under our tabletop protocol (y-axis). The points cluster close to the diagonal, indicating that the two evaluation protocols give similar success-rate estimates. Reference lines mark GOTT’s tabletop success rates of 0.74 (XHand), 0.75 (Sharpa), and 0.87 (Allegro), which lie well above BODex’s results. Even when BODex grasps are evaluated under its original floating-hand protocol, GOTT outperforms BODex on every embodiment, confirming that the gap in Table 1 of the main paper is not an artifact of the evaluation choice.

E.4. Protocol Comparison with DRO

DRO [53] evaluates its grasps in Isaac Gym, where a controller closes the hand around the object. The grasp is then perturbed by sequential forces along the positive and negative directions of each coordinate axis, with a magnitude corresponding to an acceleration of 0.5 m/s^2 (i.e., $F = m_{\text{obj}}(0.5 \text{ m/s}^2)$). A grasp is successful if the object’s displacement stays below 2 cm. Our protocol differs in two ways. First, we require the grasp to lift the object against gravity rather than resist a brief perturbation. The gravitational load $F_g = m_{\text{obj}}(9.8 \text{ m/s}^2)$ is roughly $20\times$ the perturbation force F used in DRO’s protocol and is sustained throughout the trial. Second, our protocol places the object on a table from the start, whereas DRO’s grasp synthesis does not account for table collisions, which limits its performance under our protocol.

E.5. DRO Baseline Variants

Our main table (Table 1) uses DRO’s [53] released checkpoint. DRO is trained across embodiments and transfers zero-shot to new hands given their URDFs, but XHand and Sharpa Wave are not part of its training set. For a fair comparison, we additionally fine-tune or retrain DRO on our BODex-generated YCB grasp set. Specifically, we run BODex’s tabletop-aware bilevel optimization from the *same* initial hand configurations used by GOTT and keep the 4,000 grasps that pass BODex’s own validity check as the training set. All variants are evaluated under our tabletop protocol (Table 13). All fine-tuning and retraining variants underperform the released checkpoint on average across hands. We attribute this to our smaller 16-object training set with tabletop-constrained grasps, which differs from DRO’s original training distribution. We therefore keep the released checkpoint in the main table as the strongest DRO variant overall. Notably, even on the natively supported Allegro hand, DRO reaches 43.2% versus our 87.1%, indicating that the gap primarily reflects the benefit of closed-loop contact acquisition over open-loop grasp execution under the same tabletop protocol.

Table 13: **DRO variants.** Lifting success rate (%) under our tabletop protocol; best per column in bold.

DRO variant	Allegro	Sharpa	XHand	Av.
Released ckpt	43.2	47.0	31.3	40.5
Fine-tune 1 ep.	21.6	52.2	31.1	35.1
Fine-tune 5 ep.	20.0	37.9	32.0	30.0
Retrain 5 ep.	19.8	40.6	32.6	31.0
Retrain 40 ep.	18.6	35.5	31.3	28.5

F. Additional Experiments

F.1. Runtime Breakdown

Table 14 reports the per-stage runtime averaged over 10 rollouts. One-time initialization (4-view rendering, VLM-based affordance reasoning, candidate generation, IK filtering, and cuRobo planning) totals about 29.6 s. Online execution is real-time: FoundationPose runs at 45 Hz on an RTX 4090, and the contact-acquisition and contact-prediction networks take 0.752 ms and 0.085 ms per step, respectively, supporting the 10 Hz control loop.

F.2. Contact Predictor Evaluation

We evaluate the contact predictor described in Section A.7 on unseen rollouts of the frozen contact-acquisition primitive. We use the 16 training YCB objects under the same domain randomization, with 2560 episodes of 60 steps per hand.

Table 14: Per-stage runtime, averaged over 10 rollouts.

Stage	Time
4-view rendering	0.11 ± 0.01 s
VLM-based affordance reasoning	14.9 ± 0.9 s
Candidate generation	0.76 ± 0.01 s
IK filtering	3.95 ± 0.06 s
cuRobo planning ($N=10$)	9.6 ± 0.5 s
FoundationPose	45 Hz
Contact acquisition (per step)	0.752 ms
Contact predictor (per step)	0.085 ms

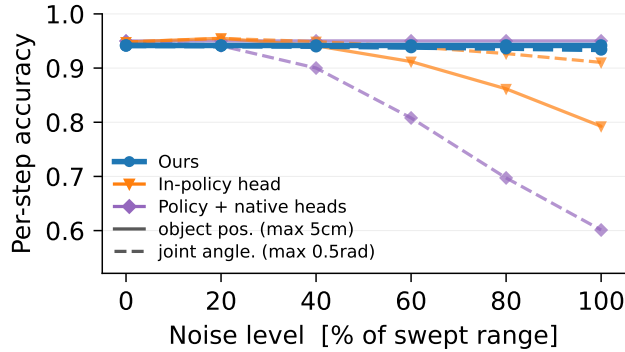


Figure 15: Contact-predictor accuracy under observation noise. Per-step accuracy of the debounced switch signal, averaged over the three hands, as the predictor input is perturbed along one axis at a time (solid: object position, 0–5 cm; dashed: joint angle, 0–0.5 rad). Our proprioceptive predictor is unaffected by object-pose noise and degrades least under joint noise.

Evaluation protocol. At each control step, `is_grasping` is set according to the per-finger contact predictions. We compare this signal against the simulator’s ground-truth grasp state. A false positive is a step at which `is_grasping` is true while the simulator indicates no grasp, and a false negative is the reverse. At the episode level, a switch is considered correct if the predictor triggers tracking while the hand is actually holding the object.

Results. At a contact-probability threshold of $\tau_c = 0.5$, the per-step false-positive rate is 0.066–0.125 and the false-negative rate is 0.036–0.047 across the three hands. Per-episode switch accuracy is 0.953–0.960, with a median lag of at most one step relative to the simulator’s switch. Switch decisions are insensitive to the threshold: switch accuracy remains 0.94–0.98 for $\tau_c \in [0.4, 0.8]$ and degrades only for $\tau_c \leq 0.3$.

Robustness to observation noise. We train a separate proprioception-only predictor because it is more robust to observation noise than a contact head coupled to the policy. To verify this, we compare against two alternatives: a head inside the policy network trained jointly with PPO from the full policy observation, and per-embodiment proprioceptive predictors trained alongside the policy instead of after it. All three are evaluated offline on the same recorded rollouts using their respective inputs and the same ground-truth labels. Noise is added only to the predictor input, one axis at a time (object position

up to 5 cm, joint angles up to 0.5 rad), while the rollouts themselves are left unchanged. As shown in Figure 15, our predictor is unaffected by object-pose noise by construction and loses at most 0.01 in accuracy under 0.5 rad joint noise, whereas the in-policy head loses on average 0.16 under object-position noise and the jointly trained heads lose on average 0.35 under joint noise.