Assembling Two Parts in One Hand

Liuao Pei[1,2,*] **Tianyue Wu**[1,2,*] **Hui Zhang**[4] **Ping Luo**[1,†] **Jie Song**[2,3,†]

[1]The University of Hong Kong
[2]The Hong Kong University of Science and Technology (Guangzhou)
[3]The Hong Kong University of Science and Technology
[4]ETH Zurich

[*]Equal contribution. The first two authors are listed in alphabetical order.
[†]Corresponding authors.

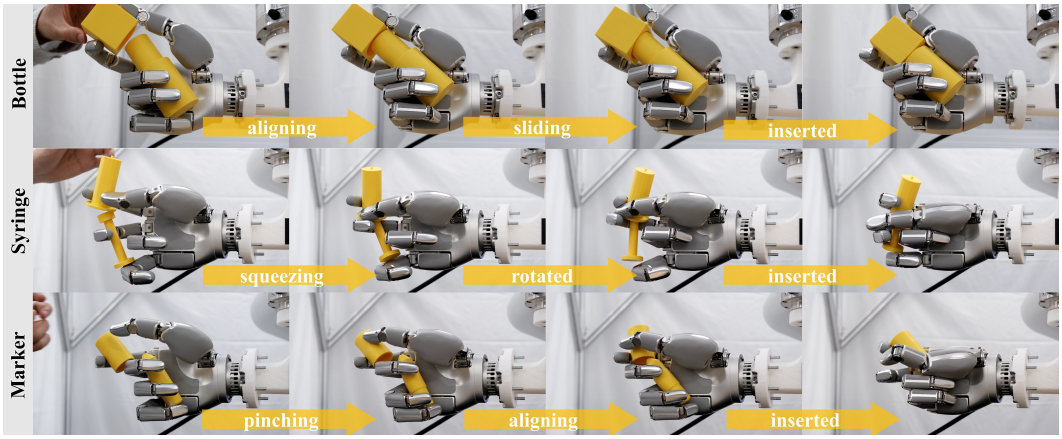

Figure 1: **Real-world in-hand assembly.** A single dexterous hand mates two objects using fine finger coordination without a second arm or external fixture, shown here on the Bottle, Syringe, and Marker tasks.

Abstract: A hallmark of human dexterity is the cooperative use of fingers, where different fingers take on distinct yet coordinated roles to accomplish fine manipulation, such as capping a pen with the hand that holds it. We study this finger-level coordination through *in-hand assembly*: mating two rigid objects within a single dexterous hand, with no second arm and no fixture. We present a reinforcement learning formulation to solve this problem in a unified framework, which is driven by a goal relative pose between the two parts. Finger coordination is shaped by a function-based auxiliary reward and regularized toward a single human reference pose, while domain randomization and a fusion of historical proprioception and object observation confer robustness to occlusion-induced estimation noise. The same recipe solves three different assembly tasks (Bottle, Syringe, and Marker). Trained purely in simulation, the policies transfer zero-shot to hardware with a single camera, demonstrating robustness to state-estimation errors caused by occlusion. Our experiments also reveal that in-hand assembly places demands on hand morphology and can serve as a benchmark for modern robotic hand systems. Videos and code are available at this URL.

Keywords: Dexterous Manipulation, In-Hand Assembly, Reinforcement Learning, Sim-to-Real

1 Introduction

While animals can also perform basic manipulation behaviors with their limbs or digits, such as pushing or pulling objects, the ability to execute complex manipulation through coordinated finger interactions is most refined in human hands [1, 2]. For instance, humans squeeze a syringe with one hand, cap a pen with the hand that holds it, or press a lid back onto a bottle held in the same hand. This cooperative use of individual fingers is common in everyday human manipulation, whereas current work on dexterous manipulation remains limited to relatively simple or repetitive finger motions and contact modes, as exemplified by pick-and-place tasks [3, 4, 5, 6] and in-hand reorientation [7, 8, 9]. These capabilities have yet to fully exploit the dexterity of modern robotic hands.

To move beyond these limited settings, we study tasks that more directly reflect and exploit the coordinative dexterity of human hands. Specifically, we focus on *in-hand assembly*: tasks in which a single multi-fingered hand simultaneously manipulates two objects to mate them and operates the assembly under articulation constraints (Fig. 1). In-hand assembly tasks, however, are inherently challenging in several aspects. First, the tasks are contact-rich and control-intensive, requiring forceful yet precise coordination across four or five fingers. Such intricate multi-finger motions are difficult to obtain through teleoperation [10] or finger-by-finger kinesthetic demonstration [11]. Second, assembly tasks exhibit diversity in object geometry, mating mechanisms, and target motions. Together with the large number of degrees of freedom in the control space, this often leads to task-specific formulations and optimization objectives, posing a major scalability challenge. Third, assembly requires precise alignment, while occlusion-induced errors in object state estimation pose significant challenges to control robustness.

To address these challenges, we develop a policy learning approach based on reinforcement learning (RL) in simulation to automatically explore solutions. From the perspective of relative object state goal reaching, we abstract a shared task formulation that captures reusable structure across assembly tasks, with different roles assigned to different fingers, such as object translocation and compliant support, triggered by a function-relevant reward. We further propose reference snapshots from human manipulation to constrain exploration. To enable robust closed-loop execution under occlusion, we apply observation domain randomization during training, encouraging the policy to implicitly calibrate noisy estimates through proprioceptive feedback.

Our contributions are threefold: (1) **In-hand assembly as a dexterity benchmark**. We introduce in-hand assembly as a challenging dexterous manipulation setting. By requiring one hand to coordinate multiple fingers across two interacting objects, it provides a task family for evaluating coordinative dexterity in robotic manipulation systems. (2) **Systematic solution**. We present a systematic solution for in-hand assembly that integrates simulation-based RL, finger function rewards, and human kinematic priors, which is unified across tasks. To our knowledge, these components enable the first fixture-free assembly demonstration through a general-purpose anthropomorphic hand. (3) **Real-world effectiveness**. The system performs real-world in-hand assembly using only stereo vision from a single camera and historical proprioception, demonstrating effectiveness under state-estimation noise and unexpected disturbance.

2 Related Work

2.1 Dexterous Manipulation with Multi-Fingered Hands

Most work on multi-fingered hands centers on grasping, either synthesizing large-scale datasets [3, 12, 13] or training policies to grasp novel objects [14, 15, 16, 17]. Such grasps often reduce the hand to enveloping a single object with all fingers, which a low-DoF gripper can also achieve [18]. A smaller body of work grasps several objects with one hand [19, 20, 21], but partitions the fingers into subsets that each enclose a separate object, acting as independent grippers that co-hold the objects rather than coordinating them, leaving the coordinative dexterity limited. Another line uses sim-to-real RL [22, 23] for in-hand manipulation of a single object, typically on easy-to-formulate skills such as in-hand rotation [7, 8, 9, 24, 25, 26] or translation [27]. Others learn from teleoperated demonstrations [11, 28, 29, 30] or human motion retargeting [31, 32], but the embodiment gap between human and robot hands hinders agile, highly dexterous skills with high precision. Overall,

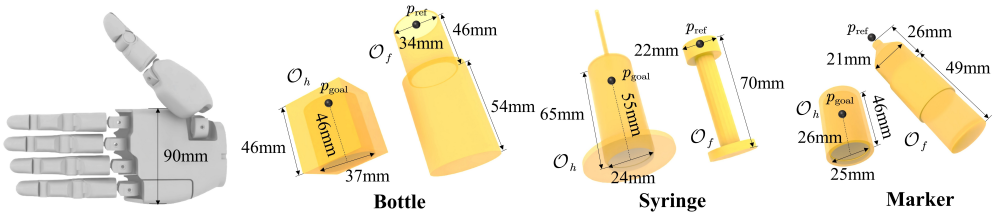

Figure 2: **Task illustration with key geometric parameters.** This figure includes the semi-transparent parts of three task instances, key geometric parameters, and the positions of the reference point p_{ref} and goal point p_{goal} to define a goal-reaching reward (see §3.4).

existing systems still do not capture the human-like dexterity of precisely coordinating multiple objects within a single hand. In contrast, we use multi-object in-hand assembly to expose this gap and probe the boundary of machine dexterity with modern multi-fingered hands.

2.2 Robotic Assembly

Assembly is a long-standing problem in robotic manipulation, with insertion (peg-in-hole) as its canonical instance. Current learning-based methods show that sim-to-real RL can acquire robust contact-rich insertion policies [33, 34, 35, 36]. Almost all of this work, however, uses a parallel-jaw gripper on a robot arm: the gripper rigidly clamps one part while the other is held by a fixture or a second arm. Lee et al. [37] perform keyhole peg-in-hole with a dual-arm system equipped with robotic hands in which the fingers only grasp and pre-rotate each part. More broadly, two-object contact-rich skills with multi-fingered hands are performed bimanually, e.g., twisting lids off bottles with one hand stabilizing and the other turning [38], and target disassembly rather than the harder mating of parts. Triyonoputro et al. [39] demonstrate in-hand assembly with a specialized double-jaw mechanism. We instead study fixture-free assembly through coordinated fingers of a general-purpose anthropomorphic hand, providing a benchmark for its intrinsic dexterity.

3 Method

We formulate *in-hand assembly* as a reinforcement learning problem (§3.1), and introduce three tasks for this setting (§3.2). Based on this, we design our unified observation & action spaces (§3.3), and reward structure (§3.4). We also involve human manipulation snapshots for initialization (§3.5), and apply domain randomization during training to improve robustness (Appx. §A). The simulation setup especially for contact-rich assembly [33], RL implementation details, and policy architectures are postponed to Appx. A.

3.1 Problem Formulation

Let $\mathcal{O}_h$ and $\mathcal{O}_f$ be the two rigid bodies to be mated, and let $\mathcal{H}$ be a *single* dexterous hand with n_q revolute joints. At every control step, the hand must drive $\mathcal{O}_h$ into a *goal relative pose* with respect to $\mathcal{O}_f$ that satisfies a task-specific geometric mating constraint. The names (held versus fixed) follow the assembly literature [34], whereas no second arm, table fixture, or vise is available; both objects are supported by the same hand.

We partition $\mathcal{H}$'s fingers into two role sets and regularize their motion using separate reward components (see §3.4) that softly constrain which fingers contact which object. The resulting behavioral specialization (reported in §4) is: the held set $\mathcal{F}_h = \{\text{thumb}, \text{index}\}$ manipulates $\mathcal{O}_h$ at its top and provides the fine alignment and goal-pose-reaching motion. The fixed set $\mathcal{F}_f = \{\text{middle}, \text{ring}, \text{little}\}$ cages $\mathcal{O}_f$ and resists the reaction wrenches generated by $\mathcal{F}_h$, while also assisting with the alignment and assembly, e.g., the middle and ring fingers might fine-tune the pose of $\mathcal{O}_f$, and the pinky finger can assist with insertion.

Casting this as a partially observable Markov decision process (POMDP) $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, \gamma)$, we seek a recurrent policy $\pi_\theta(a_t \mid o_{\le t})$ that maximizes $\mathbb{E}_\pi\left[\sum_{t=0}^{T_{\max}} \gamma^t R_t\right]$ subject to joint-limit and contact-feasibility constraints.

3.2 Task Instances

We have instantiated in-hand assembly into three tasks: (1) Bottle cap and bottle assembly (Bottle); (2) Syringe insertion (Syringe); (3) Marker cap fitting (Marker), as shown in Fig. 2. The primary differences among these three assembly tasks lie in the parts' geometries, assembly hand pose, and tolerances. For instance, the parts in the Bottle task are relatively large, which can lead to unstable grasping, and adjusting the bottle's orientation (often necessary) requires the involvement of each finger's motion. In the Syringe task, the plunger requires contact with the back of the ring finger and the involvement of the pinky finger for insertion, while, in the Marker task, the hand causes extensive occlusion, placing demands on the robustness of the state estimator and the policy against estimation errors. The assembly types for all three tasks are similar to those described in previous assembly literature [34, 37], specifically plug-in assembly. In the future, we plan to expand to other types of assembly, such as threading and friction-fit assemblies; however, these will require more accurate simulation [40].

3.3 Observation and Action Spaces

Observation. The policy reads a 34-dimensional vector

$$o_t \;=\; \left[\, p_t^{(h)} \,\middle|\, \hat{z}_t^{(h)} \,\middle|\, p_t^{(f)} \,\middle|\, \hat{z}_t^{(f)} \,\middle|\, q_t \,\right] \;\in\; \mathbb{R}^{3+3+3+3+22}, \tag{1}$$

where $p^{(\cdot)} \in \mathbb{R}^3$ is the object centroid in the hand-base frame, $\hat{z}^{(\cdot)} \in S^2$ is the unit vector representing the object's body-frame z-axis, which is the object's axis of rotational symmetry, in the world coordinates, and $q_t \in \mathbb{R}^{22}$ is the vector of measured joint angles. If we randomize hand tilt during training, the observation space is extended to include ϕ and θ, the roll and pitch angles of the hand in the world frame.

We note that the observation space consists solely of the incomplete state of the object derived from visual state estimation and the robot's proprioception; it does not include any higher-order quantities such as joint velocities or contact force signals. Since most of the components being assembled are rotationally symmetric about their body z-axis, we omit the estimation of this degree of freedom. The policy is represented as a recurrent neural network (RNN) to tackle the partial observability.

Action. The action $a_t \in [-1, 1]^{22}$ is a normalized joint-position-target delta applied to a stiff implicit Proportional-Derivative (PD) controller:

$$q_{t+1}^{\text{target}} \;=\; \text{clip}(q_t + \alpha\, a_t,\; q_{\min},\; q_{\max}), \quad \alpha = 0.1 \text{ rad}, \tag{2}$$

where q_t is the current PD joint-position target, and $q_{\min}$ and $q_{\max}$ represent the lower and upper joint limits, respectively. We control at 15 Hz, and this action structure results in naturally smooth absolute joint angles.

Real-World Object Pose Tracking. At deployment, we recover the 6-DoF poses of $\mathcal{O}_h$ and $\mathcal{O}_f$ from an RGB-D stream with FoundationPose [41], which registers each object mesh against the first frame and refines the pose per frame thereafter. Each pose is then reduced to the $(p, \hat{z})$ form of Eq. (1); a lightweight depth-consistency gate rejects per-frame estimates whose predicted depth disagrees with the measured depth at the projected center, which prevents the tracker from locking onto the manipulating hand when it occludes the object. More details can be found in Appx. B.

3.4 Goal-Reaching with Auxiliary Rewards

Goal-reaching reward. The main reward signals come from:

$$R_t^{\text{goal}} \;=\; 5 \cdot (\underbrace{\exp(-60 e_{xy} - 60 e_z - 5 e_\theta)}_{\in [0,1]} + \underbrace{\exp(-200 e_{xy} - 120 e_z - 30 e_\theta)}_{\in [0,1]}), \tag{3}$$

where e_{xy} and e_z are the xy-plane and z-axis distances between the reference point and the goal point illustrated by Fig. 2 in $\mathcal{O}_f$'s local frame (in meters), and $e_\theta = 1 - z_{\text{held}} \cdot z_{\text{fixed}}$, with z_{held} and z_{fixed} being the normalized direction vectors of the body z-axes of $\mathcal{O}_h$ and $\mathcal{O}_f$, respectively. This reward describes the achievement of relative spatial relationship between the two parts, which is the objective of the assembly task. We set up two additive terms to provide goal-reaching motivations at different scales.

Auxiliary reward. Active throughout the episode, it is a multiplicative combination of three normalized quality signals:

$$R_t^{\text{aux}} = \underbrace{\rho_t^{\text{pinch}}}_{\in[0,1]} \cdot \underbrace{\rho_t^{\text{grasp}}}_{\in[0,1]} \cdot \underbrace{\rho_t^{\text{pose}}}_{\in[0,1]}. \tag{4}$$

ρ_t^{pinch} counts how many of {thumb, index} are in contact with $\mathcal{O}_h$; $\rho_t^{\text{grasp}} = \exp(-20 \cdot \max_{i \in \mathcal{F}_f} d_i)$ rewards $\mathcal{F}_f$ caging $\mathcal{O}_f$; $\rho_t^{\text{pose}} = \exp(-\|q_t - q_0\|_M^2/12)$ constrains drift from a nominal grasp pose q_0 under a per-joint mask M. These rewards reflect an empirically motivated organization of the fingers: the thumb and index finger work together to manipulate the held object (a "pinch"), while the middle, ring, and pinky fingers manipulate the fixed object (a "grasp"). The pose constraint is inspired by prior work that learns from human video, where extracted references guide wrist motion and grasping pose [42, 43]. For in-hand assembly, however, we apply such references only as single-frame snapshots: the fine, intricate finger motion of in-hand manipulation is susceptible to the embodiment gap and extraction noise, so multi-frame references offer little usable guidance for finger motion. The reward scheme leaves the fine-grained finger motion to the goal-reaching reward.

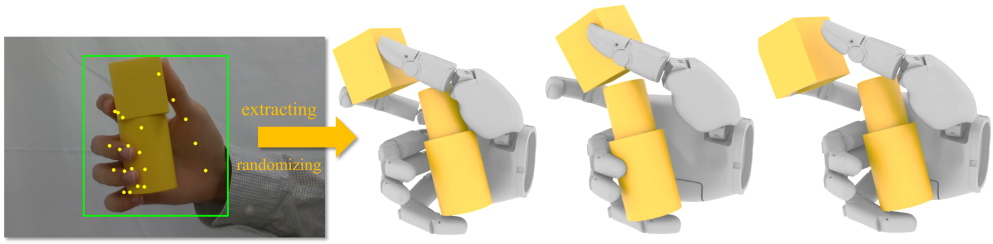

Figure 3: **State initialization from a human reference snapshot.** On the left is the human reference snapshot for the Bottle task; the three simulated states on the right are initial state examples generated based on the reference hand pose. Initialization for other tasks is postponed to Fig. 9

3.5 Snapshot as Initialization

Since the hand is anchored and cannot grasp from a tabletop, every episode must start with the objects already in hand. From a single human-demonstration snapshot of the target task, we geometrically retarget [10] to generate a robotic hand state based on the human hand pose with a 30% reduction of joint angles of the index finger to widen the opening formed by the index finger and thumb. We then apply the state estimator to extract the objects' poses. The two sources are independent, so directly composing them usually produces hand–object penetration. We resolve this and inject randomization in initial states with a settling phase: holding the hand as a fixed collider, we apply a small random perturbation force and torque to each object and step the simulator for a few steps; the perturbation is then removed, and the episode begins from the resulting state. This procedure settles the objects into randomized configurations, crucially broadening the state coverage seen during training. A more detailed randomization procedure is postponed to Appx. A. The objects' gravity is removed for the first few steps in an episode to prevent them from quickly falling. In this way, the policies learn to establish contacts with the objects in these warm-up steps.

4 Results

4.1 Experimental Setup

We use a 22-degree-of-freedom (DoF) Sharpa Wave Hand [44] in most of the simulation and real-world experiments, which is fixed on a Franka Research 3 arm [45]. The simulation experiments are conducted in IsaacSim [46]. We rely on a single RealSense D435 camera [47] to perform state estimation. We send control commands to the robot via a Linux workstation with an RTX-3090 Graphics Processing Unit (GPU). See Appx. B for more details of real-world deployment.

4.2 In-Hand Assembly as a Hand Morphology Benchmark

Intuitively, in-hand assembly places stronger demands on the morphology of the robotic hand than prior in-hand manipulation tasks, which can be accomplished by a wider variety of hands [9]. We test this hypothesis by transferring the same pipeline to different hands in simulation. Beyond the

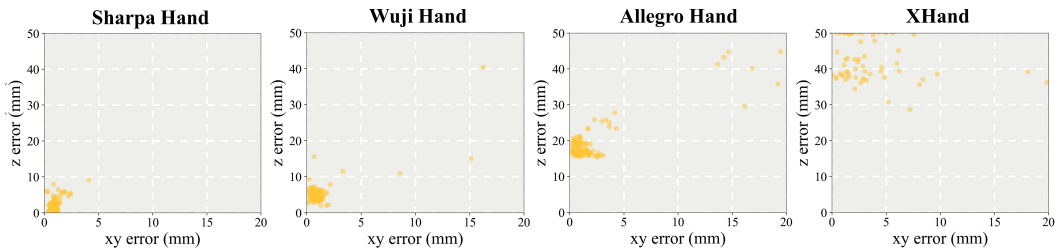

Figure 4: **Goal-reaching errors for different hand morphologies.** The yellow points in the figures are error samples with a total number of 128.

Sharpa Hand used in our main experiments, we evaluate the Wuji Hand [48], the Allegro Hand [49], and the XHand [50]. Fig. 4 reports the distribution of reference-point errors on the Syringe task. These errors are obtained from trials that do not terminate early.

From the results, we can tell that the Sharpa Hand and Wuji Hand perform best, owing to their human-like dimensions, high DoF, and wide joint ranges. The Allegro Hand has only four fingers and is larger than a typical human hand, but each finger has many degrees of freedom and a wide joint range, enabling it to align objects in the xy-plane; the lack of a fifth finger, however, prevents the deep insertion required along the z-axis. The XHand is close to human-hand size but has fewer degrees of freedom—most of its fingers lack abduction/adduction joints—and it fails to insert the objects, and the objects cannot be aligned in the xy-plane in many cases.

4.3 Simulation Ablation

We compare our policy against the following variants of our method: 1) A proprioception-only policy. This is a recurrent policy that does not incorporate object state estimation. 2) A historical input-output MLP policy (history MLP) [51]: instead of using an RNN, this policy uses 8 frames of historical policy input-output data as policy observations, implemented as an MLP. 3) A policy that does not use the finger-function-based contact reward $\rho^{\text{pinch}} \cdot \rho^{\text{grasp}}$ during training (w/o finger reward). 4) A policy that does not use the reference-pose-based reward ρ^{pose} during training (w/o pose reward). The results in Fig. 5 show the performance metric calculated using the cumulative goal-reaching reward across episodes of the models obtained at epoch 600 across 3 independent training runs. Episodes that end within 0.5s of the warmup are not included in the calculation, as they are considered to start in a poor or infeasible state. The corresponding Marker results are reported in Appendix C.

The results show that different tasks demonstrate different requirements for vision, and removing vision generally reduces task performance. For instance, removing vision significantly impacts the performance of the Bottle task. Prior work indicates that certain in-hand manipulation skills can be performed well with proprioception alone [8, 9]. Assembly, however, sometimes requires the policy to have ready access to the spatial relationship between the two objects, so having this information makes learning easier and makes it feasible to manipulate the two objects in pre-

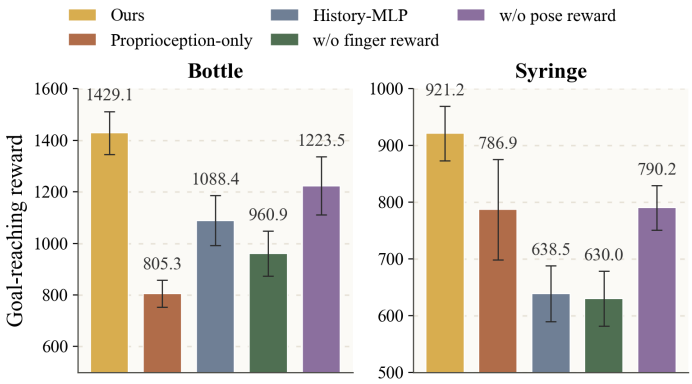

Figure 5: **Ablation study.** We compare our full policy against: (1) *proprioception-only*, which omits object state estimation; (2) *history MLP*, which replaces the LSTM with an MLP over 8 frames of input–output history; (3) *w/o finger reward*, trained without the finger-function contact reward $\rho^{\text{pinch}} \cdot \rho^{\text{grasp}}$; and (4) *w/o pose reward*, trained without the reference-pose reward ρ^{pose}.

cise relative alignment based on explicit object states. We also find that LSTM achieves better performance than stacking historical information within a fixed sample budget, implying that recurrent policies yield a better solution to the POMDP. Meanwhile, we observe that removing some components of the reward affects learning efficiency and the policies' rollout behavior at convergence. This is because the system has a high degree of operational freedom, so that many behaviors can yield goal-reaching rewards, but some of them are unnatural, do not perform well for all initial object states due to suboptimal contact modes, and thus correspond to shallower local minima of policy optimization.

4.4 Quantitative Evaluation with Estimation Error

We evaluate the goal-reaching reward (as used in the previous section) of the policies under two types of state-estimation noise at varying magnitudes. The first is zero-mean Gaussian noise, similar to the domain randomization applied during training, which models random noise and occasional outlier observations. The second adds a constant mean shift to this baseline Gaussian noise throughout the episode; such systematic offsets are plausible in practice, arising, for example, from coordinate-frame misalignment between simulation and reality or from errors in camera-parameter estimation. We note that the per-axis noise standard deviation is set to 1 cm during training.

Fig. 6 reports the results for each setting with proprioception-only baselines. For both tasks, performance remains above 60% even under relatively strong noise (e.g., a 3 cm observation standard deviation), indicating considerable robustness to out-of-distribution observation noise. For the Syringe task, however, larger offset noise drives performance below the proprioception-only baseline. Given the performance of the proprioception-only policy, we conjecture that this robustness may stem from the policy's maintenance of proprioceptive history. Appendix C reports the Marker noise analysis, and Appendix D evaluates pose-estimation errors under controlled physical occlusion.

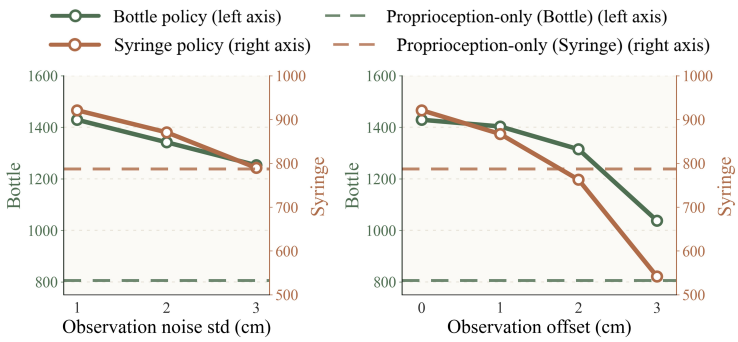

Figure 6: **Performance under different observation noise.** We report the goal-reaching reward under two noise types at varying magnitudes: zero-mean Gaussian noise (left), and the same Gaussian noise with a constant per-episode mean shift (right).

4.5 Real-World System Capabilities with a Single Camera

We conduct real-world experiments for the three tasks using only a depth camera. During the tasks, the fingers perform subtle in-hand reorientation of the objects to aid alignment and insertion, and the pinky finger is often used to increase insertion depth, as shown in Fig. 1.

Closed-loop control versus open-loop replay. To verify that closed-loop feedback is necessary, we run 20 consecutive trials comparing our closed-loop policy against open-loop replay of successful simulation trajectories. We record whether the *alignment* criterion (insertion depth over 1 cm) and the *assembly* criterion (final depth within 1 cm of the target) are met, as shown in Table 1. Open-loop replay achieves successful alignment only three times. Without feedback, the action sequences cannot compensate for sim-to-real dynamics, so early deviations accumulate, and the parts fail to

Table 1: **Closed-loop control versus open-loop replay.** Number of successful trials (out of 20) under the alignment and assembly criteria.

Method	Bottle		Syringe		Marker	
	Align	Assembly	Align	Assembly	Align	Assembly
Closed-loop (ours)	18	15	18	17	16	16
Open-loop replay	2	2	1	0	0	0

align. The closed-loop policy succeeds far more often. A main failure case in the Bottle and Marker tasks is that an unfavorable initial cap pose prevents a stable pinch, leaving the cap tilted at an unrecoverable angle or dropped. In the Bottle and Syringe tasks, certain fingers can obstruct the insertion path, a state the policy has not learned to clear.

Robustness against Perturbation. We evaluate whether the policy has the ability to recover from unexpected deviations during the assembly process by artificially altering the state of one of the manipulated objects. Fig. 7 illustrates the policy's recovery behaviors. In the Bottle task, the policy controls the index finger to correct an artificially induced misalignment of the bottle cap and realigns it without a pinch grasp. In the Syringe task, we resist the hand's grasping force to pull out the plunger, and the hand quickly restores it to the fully assembled state. These experiments validate that the policies have some robustness against unexpected state deviations.

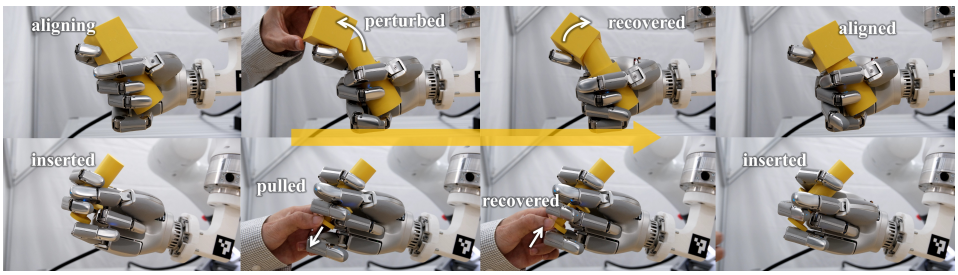

Figure 7: **Policy behaviors under unexpected perturbation.**

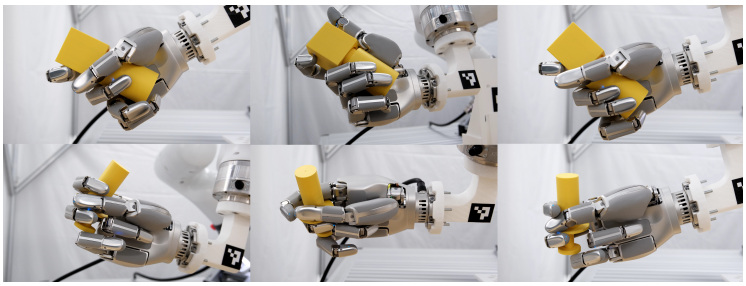

Figure 8: **In-hand assembly under different hand tilts.**

Assembly under Various Hand Tilts. In Fig. 8, we present physical experiments demonstrating assembly with a single policy trained across various wrist-tilt angles, where the wrist pose is additionally provided as a policy observation. The policy succeeds across a range of tilt angles, showing that a single policy can adapt its finger coordination to the changing relative direction of gravity. However, this experiment reveals a limitation of our method: current rigid-body simulators [52] cannot model planar (patch) contacts, so a pinch grasp degenerates into two contact points and becomes unstable in simulation [53]. This makes it hard for the policy to reliably pinch-control $\mathcal{O}_h$ at certain tilt angles, e.g., when the object's long axis starts near horizontal. Soft finger pads alleviate the instability in the real world, but introduce a sim-to-real gap. Consequently, the policy cannot learn decisive pinch behaviors in these configurations during simulation, leading to failures in real-world deployment.

5 Limitations and Future Work

A first limitation is that we study only a fragment of the in-hand manipulation workflow, rather than the full pipeline from grasping multiple objects off the table and adjusting their orientation for the subsequent assembly. The current isolation of in-hand manipulation requires artificial supporting force in some experiments until the policies engage, which may introduce factors that are difficult to model or quantify. However, a full workflow becomes long-horizon and involves skills, including multi-object grasping [19, 20] and reorienting objects from their initial placement to the desired in-hand pose [54], which remain open challenges. Second, we only currently include three tasks. In

the future, we aim to implement more general part geometries and assembly types, such as threading and friction-fit, in the current suite to expand this work into a systematic benchmark for hardware and algorithms of in-hand manipulation.

6 Conclusion

We introduced in-hand assembly, where a single dexterous hand must both stabilize and assemble objects without external fixtures or a second arm. With its requirement for intricate contact transitions and manipulation precision, this kind of task can serve as a benchmark for machine dexterity. We propose a principled RL solution with lightweight reward design, which distinguishes the roles between fingers and differentiates this RL task from other in-hand manipulation tasks. We validate its real-world effectiveness across three different tasks. These results suggest that anthropomorphic dexterity is not merely a matter of many degrees of freedom, but can be realized through human-like functional finger specialization and coordinated contact use, moving robotic hands toward human-level multi-functional dexterous manipulation.

References

[1] M. W. Marzke. Precision grips, hand morphology, and tools. *American Journal of Physical Anthropology: The Official Publication of the American Association of Physical Anthropologists*, 102(1):91–110, 1997.

[2] T. Feix, T. L. Kivell, E. Pouydebat, and A. M. Dollar. Estimating thumb-index finger precision grip and manipulation potential in extant and fossil primates. *Journal of the Royal Society Interface*, 12(106):20150176, 2015. doi:10.1098/rsif.2015.0176.

[3] R. Wang, J. Zhang, J. Chen, Y. Xu, P. Li, T. Liu, and H. Wang. Dexgraspnet: A large-scale robotic dexterous grasp dataset for general objects based on simulation. *arXiv preprint arXiv:2210.02697*, 2022.

[4] Y. Qin, B. Huang, Z.-H. Yin, H. Su, and X. Wang. Dexpoint: Generalizable point cloud reinforcement learning for sim-to-real dexterous manipulation. In *Conference on Robot Learning*, pages 594–605. PMLR, 2023.

[5] H. Zhang, Z. Wu, L. Huang, S. Christen, and J. Song. Robustdexgrasp: Robust dexterous grasping of general objects. *arXiv preprint arXiv:2504.05287*, 2025.

[6] T. Lin, K. Sachdev, L. Fan, J. Malik, and Y. Zhu. Sim-to-real reinforcement learning for vision-based dexterous manipulation on humanoids. *arXiv preprint arXiv:2502.20396*, 2025.

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

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

[9] X. Liu, H. Wang, and L. Yi. Dexndm: Closing the reality gap for dexterous in-hand rotation via joint-wise neural dynamics model. *arXiv preprint arXiv:2510.08556*, 2025.

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

[11] C. Chen, Z. Yu, H. Choi, M. Cutkosky, and J. Bohg. Dexforce: Extracting force-informed actions from kinesthetic demonstrations for dexterous manipulation. *IEEE Robotics and Automation Letters*, 2025.

[12] H. Zhang, S. Christen, Z. Fan, O. Hilliges, and J. Song. GraspXL: Generating grasping motions for diverse objects at scale. In *European Conference on Computer Vision (ECCV)*, 2024.

[13] J. Ye, K. Wang, C. Yuan, R. Yang, Y. Li, J. Zhu, Y. Qin, X. Zou, and X. Wang. Dex1b: Learning with 1b demonstrations for dexterous manipulation. *arXiv preprint arXiv:2506.17198*, 2025.

[14] T. G. W. Lum, M. Matak, V. Makoviychuk, A. Handa, A. Allshire, T. Hermans, N. D. Ratliff, and K. V. Wyk. DextrAH-g: Pixels-to-action dexterous arm-hand grasping with geometric fabrics. In *8th Annual Conference on Robot Learning*, 2024. URL `https://openreview.net/forum?id=S2Jwb0i7HN`.

[15] L. Huang, H. Zhang, Z. Wu, S. Christen, and J. Song. Fungrasp: functional grasping for diverse dexterous hands. *IEEE Robotics and Automation Letters*, 2025.

[16] Z. Chen, Q. Yan, Y. Chen, T. Wu, J. Zhang, Z. Ding, J. Li, Y. Yang, and H. Dong. Clutterdexgrasp: A sim-to-real system for general dexterous grasping in cluttered scenes. In *Conference on Robot Learning*, 2025.

[17] Y. Zhong, X. Huang, R. Li, C. Zhang, Z. Chen, T. Guan, F. Zeng, K. N. Lui, Y. Ye, Y. Liang, et al. Dexgraspvla: A vision-language-action framework towards general dexterous grasping. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 18836–18844, 2026.

[18] H.-S. Fang, C. Wang, H. Fang, M. Gou, J. Liu, H. Yan, W. Liu, Y. Xie, and C. Lu. Anygrasp: Robust and efficient grasp perception in spatial and temporal domains. *IEEE Transactions on Robotics (T-RO)*, 2023.

[19] S. He, Z. Shangguan, K. Wang, Y. Gu, Y. Fu, Y. Fu, and D. Seita. Sequential multi-object grasping with one dexterous hand. *arXiv preprint arXiv:2503.09078*, 2025.

[20] H. Lu, Y. Dong, Z. Weng, F. T. Pokorny, J. Lundell, and D. Kragic. Grasping a handful: Sequential multi-object dexterous grasp generation. *IEEE Robotics and Automation Letters*, 2025.

[21] H. Lu, Y. Dong, Z. Weng, F. Pokorny, J. Lundell, and D. Kragic. Grasping a handful: Sequential multi-object dexterous grasp generation. *IEEE Robotics and Automation Letters*, 2025.

[22] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

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

[24] M. Yang, C. Lu, A. Church, Y. Lin, C. Ford, H. Li, E. Psomopoulou, D. A. Barton, and N. F. Lepora. Anyrotate: Gravity-invariant in-hand object rotation with sim-to-real touch. *arXiv preprint arXiv:2405.07391*, 2024.

[25] Z.-H. Yin, B. Huang, Y. Qin, Q. Chen, and X. Wang. Rotating without seeing: Towards in-hand dexterity through touch. *arXiv preprint arXiv:2303.10880*, 2023.

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

[27] J. Yin, H. Qi, J. Malik, J. Pikul, M. Yim, and T. Hellebrekers. Learning in-hand translation using tactile skin with shear and normal force sensing. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5850–5856. IEEE, 2025.

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

[29] S. Yang, M. Liu, Y. Qin, D. Runyu, L. Jialong, X. Cheng, R. Yang, S. Yi, and X. Wang. ACE: A cross-platfrom visual-exoskeletons for low-cost dexterous teleoperation. *arXiv preprint arXiv:2408.11805*, 2024.

[30] T. Lin, Y. Zhang, Q. Li, H. Qi, B. Yi, S. Levine, and J. Malik. Learning visuotactile skills with two multifingered hands. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5637–5643. IEEE, 2025.

[31] K. Li, P. Li, T. Liu, Y. Li, and S. Huang. Maniptrans: Efficient dexterous bimanual manipulation transfer via residual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6991–7003, 2025.

[32] X. Liu, J. Adalibieke, Q. Han, Y. Qin, and L. Yi. Dextrack: Towards generalizable neural tracking control for dexterous manipulation from human references. *arXiv preprint arXiv:2502.09614*, 2025.

[33] Y. Narang, K. Storey, I. Akinola, M. Macklin, P. Reist, L. Wawrzyniak, Y. Guo, A. Moravanszky, G. State, M. Lu, A. Handa, and D. Fox. Factory: Fast contact for robotic assembly. In *Robotics: Science and Systems (RSS)*, 2022. doi:10.15607/RSS.2022.XVIII.035.

[34] B. Tang, M. A. Lin, I. Akinola, A. Handa, G. S. Sukhatme, F. Ramos, D. Fox, and Y. Narang. Industreal: Transferring contact-rich assembly tasks from simulation to reality. In *Robotics: Science and Systems (RSS)*, 2023.

[35] B. Tang, I. Akinola, J. Xu, B. Wen, A. Handa, K. Van Wyk, D. Fox, G. S. Sukhatme, F. Ramos, and Y. S. Narang. Automate: Specialist and generalist assembly policies over diverse geometries. In *Robotics: Science and Systems*, 2024.

[36] Y. Tian, J. Jacob, Y. Huang, J. Zhao, E. Gu, P. Ma, A. Zhang, F. Javid, B. Romero, S. Chitta, et al. Fabrica: Dual-arm assembly of general multi-part objects via integrated planning and learning. *arXiv preprint arXiv:2506.05168*, 2025.

[37] D.-H. Lee, M.-S. Choi, H. Park, G.-R. Jang, J.-H. Park, and J.-H. Bae. Peg-in-hole assembly with dual-arm robot and dexterous robot hands. *IEEE Robotics and Automation Letters*, 7(4): 8566–8573, 2022.

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

[39] J. C. Triyonoputro, W. Wan, and K. Harada. A double jaw hand designed for multi-object assembly. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2018. doi:10.1109/HUMANOIDS.2018.8625049.

[40] E. Hsieh, W.-H. Hsieh, Y.-J. Wang, T. Lin, J. Malik, K. Sreenath, and H. Qi. Learning dexterous manipulation skills from imperfect simulations. *arXiv preprint arXiv:2512.02011*, 2025.

[41] B. Wen, W. Yang, J. Kautz, and S. Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 17868–17879, 2024.

[42] Z. Chen, S. Chen, E. Arlaud, I. Laptev, and C. Schmid. Vividex: Learning vision-based dexterous manipulation from human videos. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3336–3343. IEEE, 2025.

[43] T. G. W. Lum, O. Y. Lee, C. K. Liu, and J. Bohg. Crossing the human-robot embodiment gap with sim-to-real rl using one human demonstration. *arXiv preprint arXiv:2504.12609*, 2025.

[44] Sharpa Robotics. SharpaWave: A dexterous robotic hand. `https://www.sharpa.com/pages/wave`, 2025. 22-DoF anthropomorphic robotic hand. Online; accessed YYYY-MM-DD.

[45] Franka Robotics. Franka Research 3. `https://franka.de/franka-research-3`, 2022. 7-DoF torque-controlled robot arm. Online; accessed YYYY-MM-DD.

[46] NVIDIA. NVIDIA Isaac Sim. `https://github.com/isaac-sim/IsaacSim`, 2024. Open-source robotics simulation application built on NVIDIA Omniverse. Online; accessed YYYY-MM-DD.

[47] Intel. Intel RealSense Depth Camera D435. `https://www.intelrealsense.com/depth-camera-d435/`, 2018. Stereo depth camera, D400 series. Online; accessed YYYY-MM-DD.

[48] Wuji Tech. Wuji Hand: A dexterous robotic hand. https://docs.wuji.tech/docs/en/wuji-hand/latest/overview/, 2025. 20-DoF anthropomorphic robotic hand. Online; accessed YYYY-MM-DD.

[49] Wonik Robotics. Allegro Hand. https://www.allegrohand.com/, 2024. 16-DoF four-fingered robotic hand. Online; accessed YYYY-MM-DD.

[50] RobotEra. XHand1: A dexterous robotic hand. https://www.robotera.com/, 2024. 12-DoF five-fingered robotic hand. Online; accessed YYYY-MM-DD.

[51] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath. Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control. *The International Journal of Robotics Research*, 44(5):840–888, 2025.

[52] NVIDIA. NVIDIA PhysX SDK. https://github.com/NVIDIAGameWorks/PhysX, 2022. Online; accessed 1-January-2022.

[53] J. Ye, L. Wei, G. Jiang, C. Jing, X. Zou, and X. Wang. From power to precision: Learning fine-grained dexterity for multi-fingered robotic hands. *arXiv preprint arXiv:2511.13710*, 2025.

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

[55] M. e. a. Mittal. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025. URL https://arxiv.org/abs/2511.04831.

[56] D. Makoviichuk and V. Makoviychuk. rl-games: A high-performance framework for reinforcement learning. https://github.com/Denys88/rl_games, May 2021.

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

A Training Implementation

In this section, we provide a detailed overview of the training setup omitted from the main text, including simulation setup, implementation details for object initialization, policy representation, RL algorithm hyperparameters, and domain randomization.

Simulation setup. We use IsaacSim [46] for simulation-based training. We set up the simulator for efficient contact-rich simulation largely following practices in [33], which is a PhysX-based [52] implementation for simulating assembly tasks. In particular, we use signed distance function (SDF)-based collisions for the meshes in simulation; the contacts are generated with a contact patch for number reduction and solved with a Gauss-Seidel solver.

The physics timestep is $1/120\,\mathrm{s}$, and the scene uses 16 position iterations and one velocity iteration for rigid-object contacts. The fixed-base hand articulation uses a larger position-iteration budget, 32 position iterations, and one velocity iteration, which improves stability during multi-finger contact. For task objects, the contact offset is $5\,\mathrm{mm}$ and the rest offset is zero. The friction offset threshold is $10\,\mathrm{mm}$ and the friction correlation distance is $6.25\,\mathrm{mm}$, which control friction patch generation and merging.

Initial object state construction implementation Instead of directly teleporting the objects to randomized poses that can cause interpenetration with the hand, we apply a short reset-only wrench perturbation. The procedure starts from the nominal state: the hand is written to its reset joint configuration, $\mathcal{O}_f$ and $\mathcal{O}_h$ are placed at their task-specific poses extracted from the reference snapshot, as described in §3.5. For each asset, we set a bounded local-frame translational displacement Δp and a bounded local-frame rotation vector $\Delta\theta$, which represent expected maximum pose perturbation around the nominal object states. The perturbation is not applied by directly overwriting the object pose. Instead, it is converted into an external force and torque over a short reset window of $N = 2$ physics steps. With simulator timestep Δt, the implementation uses

$$c = \frac{2}{\Delta t^2 N(N+1)}$$

as the acceleration coefficient for the reset perturbation. For an object with mass m, the translational force is

$$F^W = R(\bar{q})\,\Delta p\; mc,$$

where $R(\bar{q})$ maps the sampled local displacement into the world frame. The rotational perturbation uses an approximate diagonal inertia model for the object to be manipulated. If I is the diagonal inertia estimate, the local torque is

$$\tau^L = I\,\Delta\theta\, c,$$

and it is then rotated into the world frame before being applied to the object. During the reset-only wrench rollout, the robot is repeatedly rewritten to its reset joint state and root state. Thus, the hand acts as a static collision body while the objects settle under the sampled perturbation. After the short simulation window, the external wrench is cleared. The resulting object poses are clamped relative to their nominal states so that neither translation nor rotation exceeds the configured bounds. Finally, both object root velocities are set to zero.

Termination condition. Each task terminates after a certain number of steps. We use task-dependent episode horizons in simulation, where the Bottle task's episode length is 20s, the Syringe's is 10s, and the Marker's is 20s. Early termination occurs when one object moves 0.5m away from the hand, e.g., falling from the hand.

Policy representation. The policy is a recurrent Gaussian whose backbone is a two-layer LayerNorm-LSTM with 1024 hidden units, followed by a $[512, 128, 64]$ ELU MLP that emits the action mean. A separate head shares the same backbone to estimate the discounted value. The history MLP baseline is implemented as an $[1024, 512, 128, 64]$ ELU MLP.

RL Setup. Policies are trained with the Isaac Lab wrapper [55] for `rl_games` [56], where the Proximal Policy Optimization (PPO) [57] implementation is used. Task variants with randomized hand tilt use four times as many parallel environments as those with fixed hand tilt; their actor and central-value minibatch sizes are scaled by four compared to the fixed tilt setup to keep the per-update minibatch ratio comparable. The detailed hyperparameters are listed in Table 2.

Table 2: Key RL training parameters. Entries separated by "/" denote the fixed hand tilt task and randomized hand tilt settings, respectively.

Parameter	Value
Parallel environments	256 / 1024
Rollout horizon	128 steps
Samples per rollout	32768 / 131072
PPO minibatch size	2048 / 8192
Central critic minibatch size	512 / 2048
Optimization epochs per rollout	4
Recurrent sequence length	128
Discount factor γ	0.995
GAE parameter λ	0.95
Learning rate	1.0×10^{-4}
Learning-rate schedule	adaptive KL
KL threshold	0.008
PPO clip range	0.2
Entropy coefficient	0.0
Critic loss coefficient	2.0
Gradient-norm clipping	1.0
Bounds loss coefficient	1.0×10^{-4}
Input/value normalization	enabled
Mixed precision	enabled

Domain randomization. We model observation errors, including those from state estimation and joint angle reading, and actuation noise by domain randomization. We apply a substantial degree of randomization to the properties of the object, such as mass and friction, while simultaneously introducing randomization into the properties of the manipulator, such as control stiffness and damping. Some variables are sampled once at reset and remain constant throughout the episode; these represent persistent physical uncertainties. Other variables are resampled at every policy step; these represent sensing or command noise. A third group is event-driven: the perturbation is triggered intermittently within an episode rather than being constant or continuously resampled. Table 3 summarizes the components of domain randomization and their ranges.

B Real-World Deployment

Robotic hand deployment procedure. In our experiments, we first fixed the wrist of the robotic hand and set the task-specific initial joint angles described in §3.5, then manually placed $\mathcal{O}_f$ between the middle, ring, and pinky fingers, and placed $\mathcal{O}_h$ between the index finger and the thumb. Since these two static fingers often cannot provide a stable pinch grip, the human continues to hold the object still until neural network control is activated, as shown in the first row of Fig. 1. In our future work, we plan to develop a fully automated process (starting with object grasping), as mentioned in §5, to address the issue of objects needing to be manually positioned at the start.

During autonomous manipulation, joint angles (proprioception) are obtained directly from the hand SDK at 500 Hz and decimated to the control rate (15 Hz). Policy actions are clipped, slew-limited to 0.12 rad per step, and interpolated to 60 Hz before the hand's position controller. A tracking-error-driven stall-relax loop monitors per-joint commanded vs. measured position and slightly relaxes targets that exceed a stall threshold, keeping motor currents below thermal limits during long,

Table 3: Domain randomization applied during training. Per-step parameters are resampled at each policy step; per-episode parameters are fixed within an episode and resampled at reset.

Parameter	Distribution	Cadence
Joint-position observation noise	$\sigma = 0.01$ rad	per-step
Joint-velocity observation noise	$\sigma = 0.10$ rad/s	per-step
Object position observation noise	$\sigma = 0.01$ m	per-step
Object orientation observation noise	$\sigma = 0.05$ (quat. components)	per-step
Action noise / delay	$\sigma = 0.02$ / $\{0, 1, 2, 3\}$ steps	per-step
Gravity magnitude / tilt	$\mathcal{U}[9.51, 10.11]$ m/s^2 / $\sigma = 0.05$ rad	per-step bias
Surface friction (per material)	$\mathcal{U}[0.5, 2.0]\times$ nominal	per-episode
Object mass	$\mathcal{U}[0.05, 0.15]$ kg (absolute)	per-episode
Joint stiffness / damping	$\mathcal{U}[0.5, 2.0]\times$ USD value	per-episode
Restitution	$\mathcal{U}[0.0, 0.5]$	per-episode
Center-of-mass offset	$\mathcal{U}[\pm 0.01\text{ m}]^3$	per-episode (epoch ≥ 600)
Warmup steps	$\mathcal{U}[7, 12] steps$	per-episode
External disturbance force	random direction, $\mathcal{U}[0, 0.3]$ N, interval $\mathcal{U}\{20, \ldots, 60\}$	event-driven

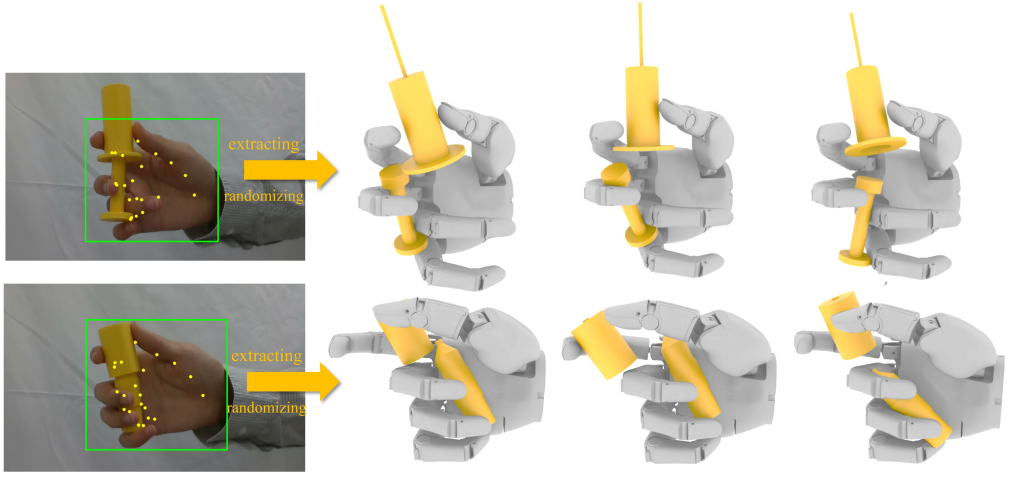

Figure 9: **Supplementary demonstration of state initialization from a human reference snapshot.** On the left is the human reference snapshot for the Bottle task; the three simulated states on the right are examples of random initial states generated based on the reference hand pose.

contact-rich grasps. Otherwise, jamming between objects caused by manipulating two objects simultaneously would lead to rapid overheating of the robotic hand and damage to the hardware.

Camera calibration. For real-world deployment, we decouple the persistent hand-fixture geometry from the per-run camera extrinsics. The fiducial marker, as shown in Fig. 10, is mounted on the hand bracket rather than placed at the task-frame origin. At the beginning of each run, the RGB-D camera observes this marker and estimates $T_{\text{marker}\leftarrow\text{cam}}$ over a short calibration window of 10 frames. We use sub-pixel ArUco corner refinement, the planar-marker PnP solution, and robust aggregation of the detected marker poses to reduce frame-level corner noise. The live marker-camera transform is then composed with a bracket calibration exported from the CAD model, $T_{\text{base}\leftarrow\text{marker}}$, and the simulator task-frame transform, $T_{\text{task}\leftarrow\text{base}}$:

$$T_{\text{task}\leftarrow\text{obj}} = T_{\text{task}\leftarrow\text{base}}\, T_{\text{base}\leftarrow\text{marker}}\, T_{\text{marker}\leftarrow\text{cam}}\, T_{\text{cam}\leftarrow\text{obj}}.$$

The policy therefore receives object poses in the same task-frame convention as simulation, while the camera may be repositioned between trials as long as the bracket marker is visible during startup. At the same startup stage, when policies trained with variable wrist pose are used, a second reference marker on the table is used to infer the current wrist orientation.

Pose-tracking stabilization. Object tracking runs on a 640×480, 30 Hz RGB-D stream. The two object meshes are first registered from color-segmented masks, with 10 refinement iterations at initialization and 2 refinement iterations in subsequent tracking steps. Since our objects are close to rotationally symmetric, we initialize the mesh orientation with a consistent upright convention and pass to the policy only the centroid and the body z-axis, rather than relying on the poorly constrained yaw angle. Stability during manipulation is enforced by rejecting physically implausible updates before they reach the policy. First, the RealSense depth stream is spatially and temporally filtered. Second, a depth-consistency gate rejects a tracked pose if the predicted object-center depth differs from the measured depth by more than 8 cm, which commonly occurs when a finger occludes the object and the tracker begins to explain the hand surface. Third, a frame-to-frame translation jump larger than 6 cm is rejected, and the internal tracker state is rolled back to the last accepted pose. Accepted poses are then smoothed with an exponential filter, using linear interpolation for translation and spherical interpolation for rotation with a coefficient of 0.7. If low-confidence tracking persists for 15 frames, the object is re-registered from color/depth contours, with contour assignment chosen by proximity to the last projected object centers to reduce identity swaps. Finally, the downstream policy consumes an atomically written latest-pose message in the task frame; transient invalid estimates are held at the most recent accepted value for at most 120 frames and marked with reduced confidence, preventing momentary occlusions from becoming discontinuous observation dropouts.

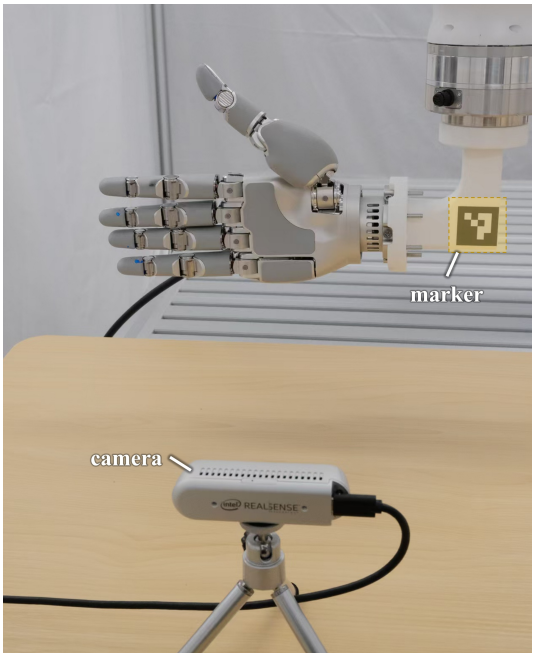

Figure 10: **The experiment scenario.**

C Marker Ablation and Noise Analyses

We extend the ablation study in Section 4.3 and the noise analysis in Section 4.4 to the Marker task. Figure 11 reports the goal-reaching reward for the full policy and the same four ablations used in the main text. We also report assembly success rates over 500 episodes per variant under domain randomization, using the same success criterion as in the real-world experiments. The full policy achieves a goal-reaching reward of 1280.1 and a success rate of 65%, compared with 338.0 and 11% for the proprioception-only policy. Removing the finger-function reward or the reference-pose reward also reduces performance.

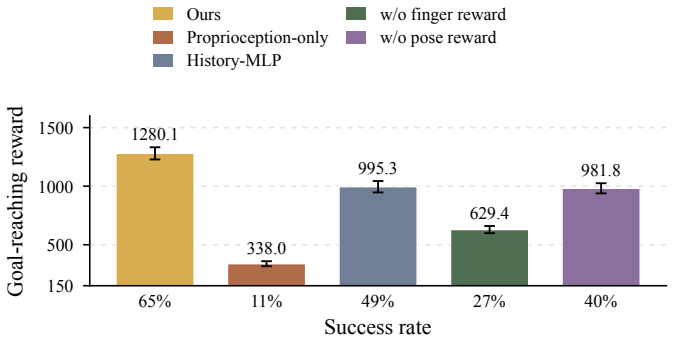

Figure 11: **Marker ablation results.** Bars show goal-reaching rewards for the same variants as in Figure 5. Reward values are annotated above the bars; assembly success rates over 500 episodes are shown below them.

Figure 12 evaluates the Marker policy under the same Gaussian and offset noise settings as in Figure 6. Performance remains stable under moderate noise and decreases as the noise magnitude increases. At a Gaussian noise standard deviation of 3 cm, the goal-reaching reward is 963.7; with a constant offset of 3 cm added to the baseline Gaussian noise, it is 670.9. Both exceed the proprioception-only baseline of 338.0. These results support robustness to moderate pose-estimation errors, while also showing degradation under larger errors.

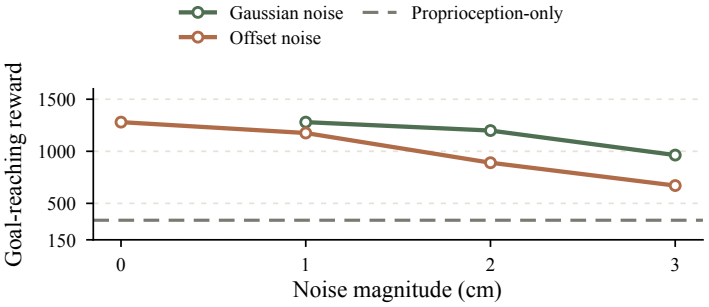

Figure 12: **Marker performance under observation noise.** The horizontal axis denotes the Gaussian noise standard deviation or the constant offset magnitude, respectively, using the noise definitions in Section 4.4. The dashed line indicates the proprioception-only baseline.

D Noise Model and Physical Occlusion

Gaussian noise models random pose-estimation errors, including occasional outliers, whereas constant offsets model systematic calibration errors. Neither directly models physical occlusion. To examine estimation errors under occlusion, we keep each object stationary, vary its visible fraction, and measure the translation error of FoundationPose relative to an unoccluded reference.

Figure 13 reports the error distributions and failure rates for both parts in each task. Most valid estimates differ from the unoccluded reference by less than 1.5 cm. However, severe occlusion can produce large errors or tracking loss. A translation error greater than 2 cm or tracking loss is counted as a failure; these failures are reported separately from the distributions of valid estimates. Thus, the small errors among valid estimates do not imply reliable tracking at all visibility levels. The Gaussian and offset noise experiments assess policy robustness to pose errors, while this controlled-occlusion experiment characterizes a physical source of those errors.

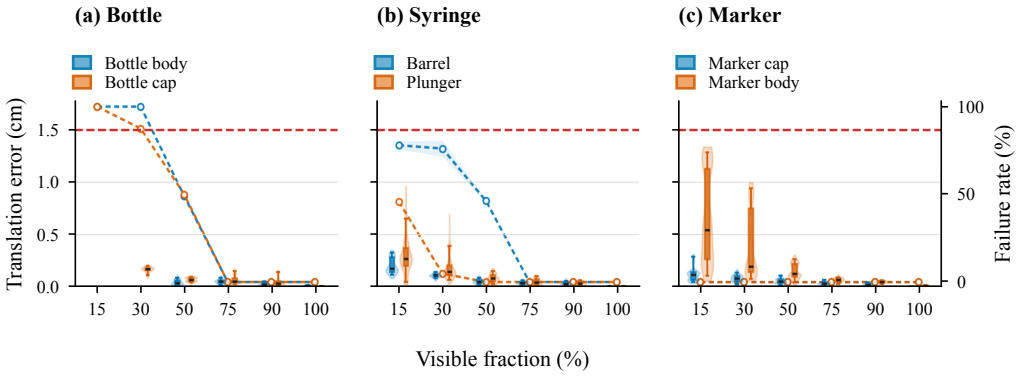

Figure 13: **Pose-estimation errors under controlled physical occlusion.** Each object remains stationary while its visible fraction varies. Blue and orange distributions show translation errors of valid estimates relative to an unoccluded reference for the two parts in each task. Dashed curves show the corresponding failure rates on the right axis; the red dashed line marks 1.5 cm.