

Play2Perfect: What Matters in Dexterous Play Pretraining for Precise Assembly?

Tyler Ga Wei Lum^{*1} Kushal Kedia^{*2} C. Karen Liu^{†1} Jeannette Bohg^{†1}

¹Stanford University ²Cornell University ^{*}Equal contribution [†]Equal advising

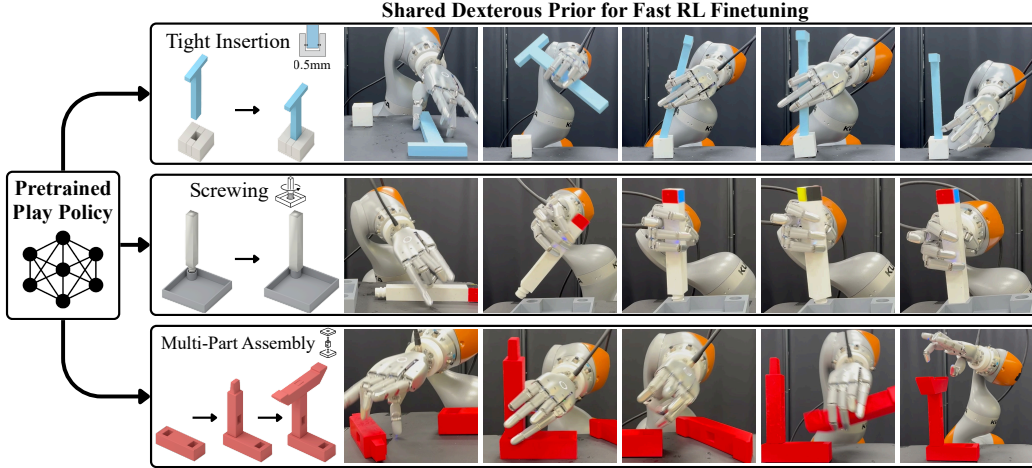


Fig. 1: **Play2Perfect Overview.** We pretrain a single goal-conditioned RL policy on task-agnostic dexterous object manipulation, producing a reusable dexterous prior. This pretrained play policy is then finetuned in sparse-reward RL environments derived from CAD designs to solve diverse contact-rich assembly tasks, including tight insertions, screwing, and multi-part assembly.

Abstract—Multi-fingered robots promise the speed and dexterity of human hands, yet challenging problems such as precise assembly have remained out of reach. These tasks are contact-rich, making data collection for imitation learning difficult, and sparse-reward, making direct exploration with reinforcement learning (RL) intractable. Consequently, prior work has made progress by structuring the problem with specialized grippers, tool attachments, and environment fixtures. In this work, we argue that before a robot can *perfect* precise assembly, it must first learn to *play*. We further ask the question: what factors in the process of learning to *play* matter for precise assembly? We propose *Play2Perfect*, an RL framework for task-agnostic pretraining through *play* on diverse objects and goals, which is then *perfected* on precise assembly. The goal of *play* is to acquire reusable manipulation priors, such as grasping, in-hand reorientation and pose reaching. Finetuning then adapts this general prior to assembly, focusing exploration on the final contact-rich, high-precision interactions needed for success. We systematically study key design choices in *play* pretraining, including object diversity, training objective, trajectory diversity, and goal precision. We show that our prior is 33x more sample-efficient than RL training from scratch, even when provided with dense, multi-stage rewards. We demonstrate zero-shot sim-to-real transfer, achieving 60% success on tight insertions with only 0.5 mm contact clearance, and over 50% success on long-horizon multi-part assembly and screwing.

I. INTRODUCTION

Robots with multi-fingered hands hold the potential to bring speed and dexterity to the diverse tasks humans perform with their hands. Yet, this flexibility comes at the cost of controlling

many degrees of freedom through contact, leaving challenging domains like precise assembly out of reach for current robot learning methods. On one hand, the contact-rich nature of assembly makes dexterous teleoperation challenging, so most imitation learning has focused on lower-precision pick-and-place tasks [58, 14, 15, 39, 7]. On the other hand, assembly is sparse-reward, defined by a part’s final pose, limiting the use of sim-to-real RL methods that require dense-reward shaping [56, 65, 12, 2, 19, 27, 13].

Prior work has made progress on assembly by adding structure to the problem. One approach is to modify the environment with custom fixtures that simplify grasping and insertion [32, 42]. Another is to modify the robot with specialized tool attachments or end-effectors that make the control problem easier [17, 44]. However, both strategies require per-assembly engineering of the hardware or environment. The use of robots with parallel-jaw grippers makes teleoperation feasible, enabling imitation learning [3] and subsequent RL fine-tuning [4]. Without teleoperation, RL methods often rely on dense, task-specific rewards [51, 52] or scripted multi-stage controllers [55]. The reliance of these approaches on parallel-jaw grippers still limits their speed and dexterity.

Our goal is to tackle hard, sparse-reward problems such as precise assembly with dexterous hands, without relying on teleoperation. The core challenge is that the sparse reward defined by the final goal configuration of an assembly part offers limited training signal for RL. Starting from a random

policy, the agent must discover grasping, in-hand reorientation, alignment, and contact-rich insertion before receiving any reward. Intuitively, before learning the hard problem of *perfecting* precise assembly, a robot should first learn the easier problem of *playing* with objects in free space. While the concept of learning from *play* has been explored previously [33, 57, 25, 23], it remains unclear what aspects of the *play* pretraining recipe matter for downstream finetuning, especially for precise assembly. In this work, we systematically study the design choices that make *play* useful for precise assembly, including object diversity, trajectory diversity, training objective, and goal precision. Across these studies, we find a consistent takeaway: *play* pretraining transfers best when it forces the robot to learn in-hand manipulation using its fingers rather than movement with a fixed grasp.

We propose *Play2Perfect*, a framework for dexterous pretraining on general objects and goals, followed by finetuning for precise assembly. In simulation, we pretrain a goal-conditioned *play* policy via RL to manipulate diverse primitive objects to random target poses, inducing a task-agnostic manipulation prior. We then construct assembly finetuning environments from assembly benchmarks [55, 20], using sparse rewards defined by the final part configuration. Across challenging assembly skills, this prior enables 33x more sample-efficient learning than RL from scratch, even when scratch policies are provided with dense, multi-stage rewards. We further demonstrate sim-to-real transfer, achieving 60% success on tight insertions with 0.5 mm clearance and over 50% success on long-horizon multi-part assembly and screwing. Our contributions are:

- A framework for precise assembly with dexterous hands that first learns a task-agnostic *play* prior on general objects and goals, then *perfects* it to new CAD-defined assembly tasks.
- A systematic study of *play* pretraining design choices that transfer to precise assembly, including object diversity, trajectory diversity, training objectives, and goal precision.

II. PLAY2PERFECT

Play2Perfect is a framework for dexterous *play* pretraining followed by sparse-reward RL finetuning for assembly. We first train a goal-conditioned policy in simulation to manipulate procedurally generated primitive objects to random poses in free-space. We then use the assembly CAD design to construct a sparse-reward RL environment for finetuning to learn precise, contact-rich goals.

A. Dexterous Play Pretraining

We first train a task-agnostic dexterous manipulation policy before specializing to any assembly task. The objective of *play* is to acquire a reusable dexterous prior by manipulating diverse objects to random goal poses. Concretely, we formulate *play* as a goal-conditioned RL problem and train a policy $\pi_\theta(s_t, o_t, g_t, \phi)$, where s_t denotes robot proprioception, $o_t, g_t \in SE(3)$ are the current and target object poses, and ϕ encodes object geometry through its 3D bounding-box dimensions. A single policy controls both the arm and hand attached

to it. Fig. 6 provides an overview of the key components of *play* pretraining that enable transfer to downstream assembly tasks. More details of the environment design, observations, and action spaces are in the Appendix.

Object Diversity. We aim to acquire broad dexterous competence across a wide range of objects, so that the resulting policy provides a useful initialization for downstream finetuning. To this end, we procedurally generate diverse primitive objects in simulation comprising cuboids and cylinders. Each object’s dimensions are sampled from a broad distribution constrained to fit within the robot hand. We also randomize physical properties by varying object densities and attaching additional masses near object ends. This induces variation in center of mass and inertia, forcing the policy to learn object-control strategies that are not tied to a single geometry or mass distribution. We use primitive shapes to enable fast, stable simulation while keeping pretraining task-agnostic.

Training Objective. Play pretraining should induce reusable dexterous skills for downstream assembly. Therefore, we train the robot to manipulate objects through a sequence of 6D goal poses: the first goal requires grasping and lifting from the table, while subsequent goals require in-hand control of object pose without drops. Each goal specifies translation and rotation: translation teaches object motion across the workspace, while rotation encourages in-hand reorientation. The play reward comprises three terms: r_{smooth} regularizes actions for smooth control, r_{grasp} encourages lifting the object above the table, and r_{goal} rewards reaching the current 6D object goal. r_{goal} includes a large sparse success bonus when $d(o_t, g_t) < \epsilon$, where $d = d_{\text{pose}}$ is a keypoint-based 6D pose distance that jointly captures translation and rotation error.

Trajectory Diversity. We randomize the goal sequence in every play episode rather than training on fixed trajectories. The first goal is sampled broadly in the robot’s workspace, while subsequent goals are sampled near the previous goal with significant rotations. This encourages learning of in-hand manipulation rather than arm movements with fixed grasps.

Goal Precision. The threshold ϵ controls the precision of the learned behavior, with $\epsilon = 1$ cm. A smaller threshold requires more accurate goal reaching, which is important for precise assembly, and forces fine object pose control.

B. RL Finetuning on Assembly Tasks

After training a general *play* policy for free-space object manipulation, we specialize it to specific assembly skills. Each assembly task is defined by the final desired part configuration and uses a sparse success reward. Finetuning adapts the pre-trained manipulation prior to the contact-rich, high-precision interactions required for assembly. In this section, we describe how we construct simulation environments and derive sparse rewards directly from the assembly CAD design.

Simulation Construction from CAD. Each assembly task is specified by a CAD design containing K parts $\mathcal{A} = \{p^i\}_{i=1}^K$ and their final assembled poses. We convert this design into a sequence of assembly steps using assembly-by-disassembly [54, 63]: starting from the completed assembly,

we identify feasible part removals and reverse this order to obtain an assembly sequence. Each step requires inserting a part p^i into a fixture f^i , defined by the parts that are already assembled. We instantiate each step as an RL environment with randomized part and fixture poses on the table.

Inferring Sparse Assembly Rewards. Assembly finetuning uses only sparse success rewards derived from the CAD-specified part configurations. For each part-fixture pair, the CAD design specifies the desired relative transform $T_{p^i}^{f^i}$ of the part p^i in the fixture frame f^i . We denote the current part pose as $p_t^i \in SE(3)$ and compute the final goal pose as $g_m^i = f_t^i T_{p,m}^f$, where $T_{p,m}^f$ is the CAD-derived part transform. For contact-rich assembly skills, we derive a small set of sparse contact goals $\mathcal{G}^i = \{g_1^i, \dots, g_M^i\}$ by reversing the assembly motion, where the final goal g_M^i corresponds to the assembled pose (see Fig. 7). For insertion, this adds an aligned pre-insertion pose at the onset of contact. For screwing, we generate poses along the thread at fixed 90° rotational offsets.

C. Training Details and Sim-to-Real Transfer

RL Algorithm and Domain Randomization. We train both play pretraining and assembly finetuning policies with Split and Aggregate Policy Gradients (SAPG) [49], which prior work [23] found to outperform PPO [41] for dexterous play. To enable sim-to-real transfer, we train all policies with domain randomization modeling action latency, proprioceptive observation delays, and noise in both current and goal object poses. Additional details are provided in the Appendix.

CAD-Based Object Pose Tracking. At deployment, we reuse the assembly CAD meshes for real-world 6D pose tracking with FoundationPose [59]. We track both the current part pose and the fixture pose. The policy runs closed-loop at 60Hz, while the object pose tracking runs at 30Hz.

III. EXPERIMENTS

We design experiments to answer the following questions:

1. Can dense rewards overcome the need for *play* pretraining?
2. Which *play* pretraining design choices matter most?
3. Can *Play2Perfect* policies transfer to the real world?

Task Design. Our robot consists of a 22-DoF Sharpa five-fingered hand mounted on a 7-DoF KUKA iiwa 14 arm. We evaluate on three tasks: 1) *Tight-Insertion*, inserting a T-shaped peg into holes with increasingly tight contact clearances; 2) *Assemble-Beam*, a multi-part beam assembly constructed from Fabrica [55]; and 3) *Screw-Leg*, screwing a furniture leg into a table fixture, constructed from FurnitureBench [20]. The original parts in Fabrica and FurnitureBench are small and designed for parallel-jaw grippers. We therefore 3D print 3×-scale parts for *Assemble-Beam* and a 3×-longer leg for *Screw-Leg*, making the tasks more suitable for dexterous hands and reliable for visual tracking under occlusion. All tasks are illustrated in Fig. 1.

Evaluation Metrics. We report two primary metrics: *Success Rate*, which measures whether the assembly part reaches its final goal configuration within a tolerance of $\epsilon = 1\text{cm}$, and *Completion Time*, which measures the average time required to

complete the task, including approach, grasping, transport, and contact-rich interaction. In simulation, we report results over 500 rollouts with randomized initial part and fixture poses. In the real world, we evaluate each task over 10 rollouts using a fixed fixture pose and randomized initial part poses.

A. Can Dense Rewards Overcome the Need for Play?

Setup. We first test whether a general dexterous prior improves RL learning compared to training from scratch. We measure success rate as a function of RL wall-clock time and compare *Play2Perfect* against two scratch baselines: *Scratch (sparse reward)*, which uses the same sparse assembly success reward as *Play2Perfect*, and *Scratch (dense reward)*, which also receives task-specific reward shaping for grasping, lifting, and tracking a sequence of 10 waypoints from the initial part pose to the fixture. In addition to the four main tasks, we include a simplified *Tight-Insertion (Fixtured)* task, where the T-peg starts propped up on a fixture.

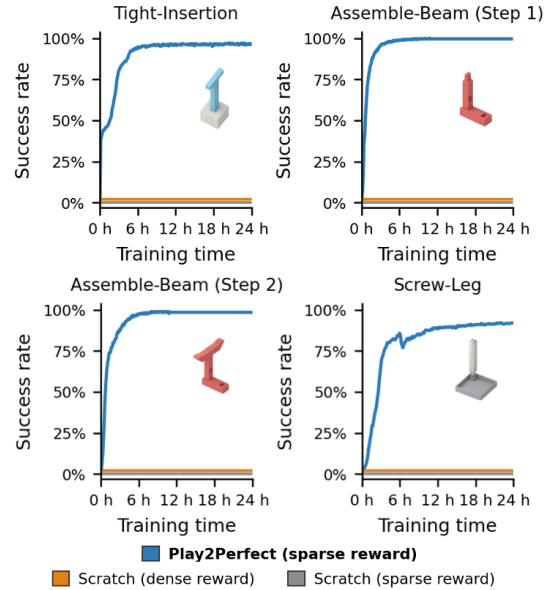


Fig. 2: **Dexterous Play Pretraining Enables Efficient Downstream Assembly Learning.** Across four assembly tasks, *Play2Perfect* rapidly learns successful policies from the shared dexterous prior, reaching high success within 2-5 hours. In contrast, training from scratch fails to make progress.

Results. Fig. 2 shows that *Play2Perfect* solves all tasks within roughly 2–5 hours of wall-clock RL training, whereas both scratch baselines produce no successful rollouts even after 24 hours. On the simplified *Tight-Insertion (Fixtured)* task, scratch training becomes feasible, as shown in Fig. 5. However, *Scratch (dense reward)* requires over 100 hours to reach near-perfect success, while *Play2Perfect* reaches the same success rate in only 4 hours, yielding a 33× speed-up. Moreover, the policy learned by *Scratch (dense reward)* remains brittle: it balances the peg using the thumb rather than forming a stable grasp. Under external force perturbations, its success rate drops to ~20%

with a 10N perturbation and eventually to 0% under larger perturbations. In contrast, *Play2Perfect* maintains over 75% success even under the largest perturbations, indicating that *play* pretraining induces a more robust manipulation strategy.

B. Which Design Choices in Play Pretraining Matter Most?

Setup. We next ablate key *play* pretraining choices and measure how they affect downstream RL finetuning on *Tight-Insertion*. Results are averaged over 3 finetuning seeds. We vary four factors: 1) *Object Diversity*: pretraining on 10, 100, or 1000 objects sampled from the same primitive distribution, with 1000 as *Play2Perfect*’s default; 2) *Training Objective*: comparing *Play2Perfect*’s full 6D goal-pose objective against *Translation-only*, which rewards only translational error along the same goal-pose sequences, and *Rotation-only*, which first lifts the object to a random 6D pose and then samples only rotational goals at a fixed position; 3) *Trajectory Diversity*: we compare using random goal trajectories (ours) against fixed sets of 10 or 100 trajectories; and 4) *Goal Precision*: varying the success threshold ϵ for goal-reaching across 1cm (ours), 5cm, and 10cm.

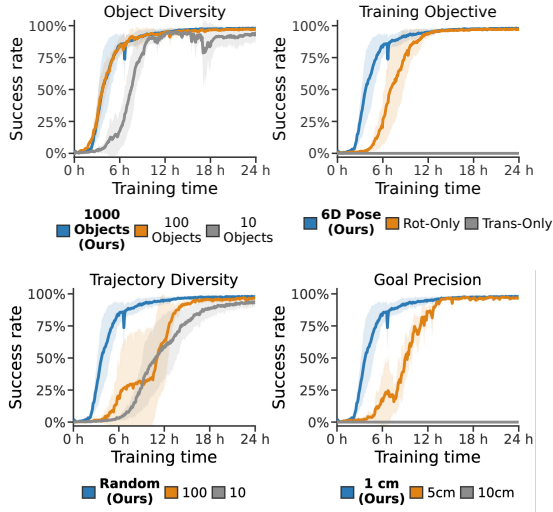


Fig. 3: **What Matters in Pretraining for Downstream Assembly Finetuning?** We vary key *play* pretraining choices and finetune each policy on *Tight-Insertion*. Curves show success rates during finetuning across 3 seeds.

Results. Fig. 3 shows that all four design choices affect downstream assembly finetuning. 1) *Object Diversity*: Increasing object diversity improves transfer, but with diminishing returns: pretraining on 100 and 1000 objects yields similar learning speed and final performance, suggesting that a moderately diverse object set is sufficient for this task. 2) *Training Objective*: Orientation control is critical. *Translation-only* pretraining learns grasping and lifting, but does not learn object orientation control, so it fails to provide the in-hand reorientation prior needed for assembly. *Rotation-only* pretraining transfers well, but learns more slowly than full 6D pose-goal pretraining, likely because it provides less practice coupling reorientation with translational object motion. 3)

Trajectory Diversity: Fixed sets of 10 and 100 trajectories perform similarly, while online random trajectories learn fastest, suggesting that broader coverage of goal-pose transitions better matches downstream assembly. 4) *Goal Precision*: Precise goals are important. A loose 10cm threshold fails to transfer because coarse goal reaching does not require accurate object-pose control. A 5cm threshold eventually learns, but more slowly than the 1cm threshold, indicating that precise *play* induces priors better matched to tight-clearance assembly.

C. Can Play2Perfect Transfer to the Real World?

Setup. We evaluate whether assembly policies finetuned in simulation can transfer zero-shot to the real world. We deploy *Play2Perfect* on *Tight-Insertion*, *Assemble-Beam*, and *Screw-Leg*, using FoundationPose [59] for pose tracking and no real-world finetuning. For *Assemble-Beam*, we evaluate each step separately to isolate per-step performance.

TABLE I: **Real-World Assembly Results.** Completion times are mean $\pm$ std over successful trials and measure the full task duration, including grasping, transport, and final contact-rich assembly. SR = Success Rate. CT = Completion Time.

	Tight-Insertion			Assemble-Beam		Screw-Leg	
	10mm	2mm	0.5mm	Step 1	Step 2	Insert	Screw
SR	10/10	9/10	6/10	8/10	7/10	7/10	5/10
CT (s)	6.8 $\pm$ 1.5	9.4 $\pm$ 1.9	11.1 $\pm$ 5.1	6.9 $\pm$ 1.9	6.4 $\pm$ 2.5	15.6 $\pm$ 2.9	

Results. Table I shows that *Play2Perfect* transfers to the real world across all three assembly tasks. On *Tight-Insertion*, the policy maintains high success as clearance tightens, achieving 10/10 at 10 mm, 9/10 at 2 mm, and 6/10 at 0.5 mm. Completion time increases from 6.8 $\pm$ 1.5 s to 11.1 $\pm$ 5.1 s as the policy performs additional local search for tighter alignment. On *Assemble-Beam*, both steps succeed reliably, with 8/10 success on Step 1 and 7/10 on Step 2, each completed in under 7 s on average. On *Screw-Leg*, *Play2Perfect* achieves 7/10 success on insertion and 5/10 on full screwing, with successful trials taking 15.6 $\pm$ 2.9 s including both phases. Completion time measures the full task duration, including approaching the object, grasping it, reorienting and transporting it, and executing the contact-rich assembly interaction. These fast execution times demonstrate the advantage of using dexterous hands for assembly, as well as the ability of RL to discover efficient manipulation strategies. See the Appendix for more analysis.

IV. CONCLUSION

We presented *Play2Perfect*, a framework for precise contact-rich assembly that finetunes a dexterous prior learned through *play*. By *play* pretraining across diverse objects, *Play2Perfect* enables fast RL on assembly tasks by adapting the prior to precise contact interactions. Our ablations show that object diversity, objectives, trajectory diversity, and goal precision all affect downstream performance. Finally, *Play2Perfect* transfers zero-shot to challenging real-world dexterous assembly tasks.

REFERENCES

- [1] Ananye Agarwal, Shagun Uppal, Kenneth Shaw, and Deepak Pathak. Dexterous functional grasping, 2023. URL <https://arxiv.org/abs/2312.02975>.
- [2] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39 (1):3–20, 2020.
- [3] Lars Ankile, Anthony Simeonov, Idan Shenfeld, and Pulkit Agrawal. Juicer: Data-efficient imitation learning for robotic assembly. *arXiv*, 2024.
- [4] Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement-residual rl for precise assembly. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 01–08. IEEE, 2025.
- [5] Miguel Arduengo, Ana Arduengo, Adrià Colomé, Joan Lobo-Prat, and Carme Torras. Human to robot whole-body motion transfer. In *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, pages 299–305, 2021. doi: 10.1109/HUMANOIDS47582.2021.9555769.
- [6] Sridhar Pandian Arunachalam, Irmak Güzey, Soumith Chintala, and Lerrel Pinto. Holo-dex: Teaching dexterity with immersive mixed reality. *arXiv preprint arXiv:2210.06463*, 2022.
- [7] Sridhar Pandian Arunachalam, Sneha Silwal, Ben Evans, and Lerrel Pinto. Dexterous imitation made easy: A learning-based framework for efficient dexterous manipulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5954–5961. IEEE, 2023.
- [8] Jose Barreiros, Andrew Beaulieu, Aditya Bhat, Rick Cory, Eric Cousineau, Hongkai Dai, Ching-Hsin Fang, Kunitatsu Hashimoto, Muhammad Zubair Irshad, Masha Itkina, et al. A careful examination of large behavior models for multitask dexterous manipulation. *Science Robotics*, 11(113):eaea6201, 2026.
- [9] Maria Bauza, Jose Enriaue Chen, Valentin Dalibard, Nimrod Gileadi, Roland Hafner, Murilo F Martins, Joss Moore, Rugile Pevceviute, Antoine Laurens, Dushyant Rao, et al. Demostart: Demonstration-led auto-curriculum applied to sim-to-real with multi-fingered robots. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6756–6763. IEEE, 2025.
- [10] Claire Chen, Zhongchun Yu, Hojung Choi, Mark Cutkosky, and Jeannette Bohg. Dexforce: Extracting force-informed actions from kinesthetic demonstrations for dexterous manipulation. *IEEE Robotics and Automation Letters*, 10(6):6416–6423, 2025.
- [11] Tao Chen, Jie Xu, and Pulkit Agrawal. A system for general in-hand object re-orientation. *Conference on Robot Learning*, 2021.
- [12] 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*, 8(84):eadc9244, 2023.
- [13] Yuanpei Chen, Chen Wang, Li Fei-Fei, and C Karen Liu. Sequential dexterity: Chaining dexterous policies for long-horizon manipulation. *arXiv preprint arXiv:2309.00987*, 2023.
- [14] Xuxin Cheng, Jialong Li, Shiqi Yang, Ge Yang, and Xiaolong Wang. Open-television: Teleoperation with immersive active visual feedback. *arXiv preprint arXiv:2407.01512*, 2024.
- [15] Runyu Ding, Yuzhe Qin, Jiyue Zhu, Chengzhe Jia, Shiqi Yang, Ruihan Yang, Xiaojuan Qi, and Xiaolong Wang. Bunny-visionpro: Real-time bimanual dexterous teleoperation for imitation learning. 2024.
- [16] Hao-Shu Fang, Branden Romero, Yichen Xie, Arthur Hu, Bo-Ruei Huang, Juan Alvarez, Matthew Kim, Gabriel Margolis, Kavya Anbarasu, Masayoshi Tomizuka, Edward Adelson, and Pulkit Agrawal. Dexop: A device for robotic transfer of dexterous human manipulation. *arXiv preprint arXiv:2509.04441*, 2025.
- [17] Huy Ha, Shubham Agrawal, and Shuran Song. Fit2Form: 3D generative model for robot gripper form design. In *Conference on Robotic Learning (CoRL)*, 2020.
- [18] Ankur Handa, Karl Van Wyk, Wei Yang, Jacky Liang, Yu-Wei Chao, Qian Wan, Stan Birchfield, Nathan Ratliff, and Dieter Fox. Dexpilot: Vision based teleoperation of dexterous robotic hand-arm system, 2019. URL <https://arxiv.org/abs/1910.03135>.
- [19] Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, et al. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5977–5984. IEEE, 2023.
- [20] Minh Heo, Youngwoon Lee, Doohyun Lee, and Joseph J. Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation. In *Robotics: Science and Systems*, 2023.
- [21] Aadithya Iyer, Zhuoran Peng, Yinlong Dai, Irmak Guzey, Siddhant Haldar, Soumith Chintala, and Lerrel Pinto. Open teach: A versatile teleoperation system for robotic manipulation. *arXiv preprint arXiv:2403.07870*, 2024.
- [22] Yunfan Jiang, Chen Wang, Ruohan Zhang, Jiajun Wu, and Li Fei-Fei. Transic: Sim-to-real policy transfer by learning from online correction. In *Conference on Robot Learning*, 2024.
- [23] Kushal Kedia, Tyler Ga Wei Lum, Jeannette Bohg, and C. Karen Liu. Simtoolreal: An object-centric policy for zero-shot dexterous tool manipulation, 2026. URL <https://arxiv.org/abs/2602.16863>.
- [24] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karam-

- cheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.
- [25] Yuxuan Kuang, Sungjae Park, Katerina Fragkiadaki, and Shubham Tulsiani. Dex4d: Task-agnostic point track policy for sim-to-real dexterous manipulation. *arXiv preprint arXiv:2602.15828*, 2026.
- [26] Kailin Li, Puhao Li, Tengyu Liu, Yuyang Li, and Siyuan Huang. Maniptrans: Efficient dexterous bimanual manipulation transfer via residual learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 6991–7003, 2025.
- [27] Toru Lin, Zhao-Heng Yin, Haozhi Qi, Pieter Abbeel, and Jitendra Malik. Twisting lids off with two hands. *arXiv preprint arXiv:2403.02338*, 2024.
- [28] Toru Lin, Yu Zhang, Qiyang Li, Haozhi Qi, Brent Yi, Sergey Levine, and Jitendra Malik. Learning visuotactile skills with two multifingered hands. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5637–5643. IEEE, 2025.
- [29] Xueyi Liu, He Wang, and Li Yi. Dexndm: Closing the reality gap for dexterous in-hand rotation via joint-wise neural dynamics model, 2025. URL <https://arxiv.org/abs/2510.08556>.
- [30] Tyler Ga Wei Lum, Martin Matak, Viktor Makoviy-chuk, 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 *8th Annual Conference on Robot Learning*, 2024. URL <https://openreview.net/forum?id=S2Jwb0i7HN>.
- [31] 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, 2025. URL <https://arxiv.org/abs/2504.12609>.
- [32] Jianlan Luo, Charles Xu, Fangchen Liu, Liam Tan, Zipeng Lin, Jeffrey Wu, Pieter Abbeel, and Sergey Levine. Fmb: a functional manipulation benchmark for generalizable robotic learning. *arXiv preprint arXiv:2401.08553*, 2024.
- [33] Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Conference on robot learning*, pages 1113–1132. Pmlr, 2020.
- [34] Zhao Mandi, Yifan Hou, Dieter Fox, Yashraj Narang, Ajay Mandlekar, and Shuran Song. Dexmachina: Functional retargeting for bimanual dexterous manipulation, 2025. URL <https://arxiv.org/abs/2505.24853>.
- [35] J Bjorck Nvidia, Fernando Castaneda, N Cherniadev, X Da, R Ding, L Fan, Y Fang, D Fox, F Hu, S Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2, 2025.
- [36] Abby O’Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.
- [37] Claudio Pacchierotti and Domenico Prattichizzo. Cutaneous/tactile haptic feedback in robotic teleoperation: Motivation, survey, and perspectives. *IEEE Transactions on Robotics*, 40:978–998, 2023.
- [38] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.
- [39] Yuzhe Qin, Wei Yang, Binghao Huang, Karl Van Wyk, Hao Su, Xiaolong Wang, Yu-Wei Chao, and Dieter Fox. Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system. *arXiv preprint arXiv:2307.04577*, 2023.
- [40] Ri-Zhao Qiu, Shiqi Yang, Xuxin Cheng, Chaitanya Chawla, Jialong Li, Tairan He, Ge Yan, David J Yoon, Ryan Hoque, Lars Paulsen, et al. Humanoid policy~ human policy. *arXiv preprint arXiv:2503.13441*, 2025.
- [41] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [42] Lin Shao, Toki Migimatsu, and Jeannette Bohg. Learning to scaffold the development of robotic manipulation skills. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5671–5677. IEEE, 2020.
- [43] Kenneth Shaw, Yulong Li, Jiahui Yang, Mohan Kumar Srirama, Ray Liu, Haoyu Xiong, Russell Mendonca, and Deepak Pathak. Bimanual dexterity for complex tasks. In *8th Annual Conference on Robot Learning*, 2024.
- [44] Haochen Shi, Huazhe Xu, Samuel Clarke, Yunzhu Li, and Jiajun Wu. Robocook: Long-horizon elasto-plastic object manipulation with diverse tools. *arXiv preprint arXiv:2306.14447*, 2023.
- [45] Zilin Si, Kevin Lee Zhang, Zeynep Temel, and Oliver Kroemer. Tilde: Teleoperation for Dexterous In-Hand Manipulation Learning with a DeltaHand. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.128.
- [46] Zilin Si, Jose Enrique Chen, M Emre Karagozler, Antonia Bronars, Jonathan Hutchinson, Thomas Lampe, Nimrod Gileadi, Taylor Howell, Stefano Saliceti, Lukasz Barczyk, et al. Exostart: Efficient learning for dexterous manipulation with sensorized exoskeleton demonstrations. *arXiv preprint arXiv:2506.11775*, 2025.
- [47] Ritvik Singh, Arthur Allshire, Ankur Handa, Nathan Ratliff, and Karl Van Wyk. Dextrah-rgb: Visuomotor policies to grasp anything with dexterous hands. *arXiv preprint arXiv:2412.01791*, 2024.
- [48] Ritvik Singh, Karl Van Wyk, Pieter Abbeel, Jitendra Malik, Nathan Ratliff, and Ankur Handa. End-to-end

- rl improves dexterous grasping policies. *arXiv preprint arXiv:2509.16434*, 2025.
- [49] Jayesh Singla, Ananye Agarwal, and Deepak Pathak. Sapg: Split and aggregate policy gradients. In *Proceedings of the 41st International Conference on Machine Learning (ICML 2024)*, Proceedings of Machine Learning Research, Vienna, Austria, July 2024. PMLR.
- [50] Aravind Sivakumar, Kenneth Shaw, and Deepak Pathak. Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube. *arXiv preprint arXiv:2202.10448*, 2022.
- [51] Bingjie Tang, Michael A. Lin, Iretiayo Akinola, Ankur Handa, Gaurav S. Sukhatme, Fabio Ramos, Dieter Fox, and Yashraj S. Narang. Industreal: Transferring contact-rich assembly tasks from simulation to reality. In *Robotics: Science and Systems*, 2023.
- [52] Bingjie Tang, Iretiayo Akinola, Jie Xu, Bowen Wen, Ankur Handa, Karl Van Wyk, Dieter Fox, Gaurav S. Sukhatme, Fabio Ramos, and Yashraj Narang. Automate: Specialist and generalist assembly policies over diverse geometries. In *Robotics: Science and Systems*, 2024.
- [53] Tony Tao, Mohan Kumar Srirama, Jason Jingzhou Liu, Kenneth Shaw, and Deepak Pathak. Dexwild: Dexterous human interactions for in-the-wild robot policies. *arXiv preprint arXiv:2505.07813*, 2025.
- [54] Yunsheng Tian, Jie Xu, Yichen Li, Jieliang Luo, Shinjiro Sueda, Hui Li, Karl DD Willis, and Wojciech Matusik. Assemble them all: Physics-based planning for generalizable assembly by disassembly. *ACM Transactions on Graphics (TOG)*, 41(6):1–11, 2022.
- [55] Yunsheng Tian, Joshua Jacob, Yijiang Huang, Jialiang Zhao, Edward Li Gu, Pingchuan Ma, Annan Zhang, Farhad Javid, Branden Romero, Sachin Chitta, Shinjiro Sueda, Hui Li, and Wojciech Matusik. Fabrica: Dual-arm assembly of general multi-part objects via integrated planning and learning. In *9th Annual Conference on Robot Learning*, 2025. URL <https://openreview.net/forum?id=aSUNzvEJIf>.
- [56] Weikang Wan, Haoran Geng, Yun Liu, Zikang Shan, Yaodong Yang, Li Yi, and He Wang. Unidexgrasp++: Improving dexterous grasping policy learning via geometry-aware curriculum and iterative generalist-specialist learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3891–3902, 2023.
- [57] Chen Wang, Linxi Fan, Jiankai Sun, Ruohan Zhang, Li Fei-Fei, Danfei Xu, Yuke Zhu, and Anima Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play. *arXiv preprint arXiv:2302.12422*, 2023.
- [58] Chen Wang, Haochen Shi, Weizhuo Wang, Ruohan Zhang, Li Fei-Fei, and C Karen Liu. Dexcap: Scalable and portable mocap data collection system for dexterous manipulation. *arXiv preprint arXiv:2403.07788*, 2024.
- [59] Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17868–17879, June 2024.
- [60] Mengda Xu, Han Zhang, Yifan Hou, Zhenjia Xu, Linxi Fan, Manuela Veloso, and Shuran Song. Dexumi: Using human hand as the universal manipulation interface for dexterous manipulation. *arXiv preprint arXiv:2505.21864*, 2025.
- [61] Ruihan Yang, Qinxu Yu, Yecheng Wu, Rui Yan, Borui Li, An-Chieh Cheng, Xueyan Zou, Yunhao Fang, Xuxin Cheng, Ri-Zhao Qiu, et al. Egovla: Learning vision-language-action models from egocentric human videos. *arXiv preprint arXiv:2507.12440*, 2025.
- [62] Jianglong Ye, Keyi Wang, Chengjing Yuan, Ruihan Yang, Yiquan Li, Jiyue Zhu, Yuzhe Qin, Xueyan Zou, and Xiaolong Wang. Dex1b: Learning with 1b demonstrations for dexterous manipulation. In *Robotics: Science and Systems (RSS)*, 2025.
- [63] Patrick Yin, Tyler Westenbroek, Zhengyu Zhang, Joshua Tran, Ignacio Dagnino, Eeshani Shilamkar, Numfor Mbiziwo-Tiapo, Simran Bagaria, Xinlei Liu, Galen Mullins, Andrey Kolobov, and Abhishek Gupta. Emergent dexterity via diverse resets and large-scale reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL <https://arxiv.org/abs/2603.15789>.
- [64] Zhao-Heng Yin, Changhao Wang, Luis Pineda, Francois Hogan, Krishna Bodduluri, Akash Sharma, Patrick Lancaster, Ishita Prasad, Mrinal Kalakrishnan, Jitendra Malik, Mike Lambeta, Tingfan Wu, Pieter Abbeel, and Mustafa Mukadam. Dexteritygen: Foundation controller for unprecedented dexterity, 2025. URL <https://arxiv.org/abs/2502.04307>.
- [65] Jialiang Zhang, Haoran Liu, Danshi Li, XinQiang Yu, Haoran Geng, Yufei Ding, Jiayi Chen, and He Wang. Dexgraspnet 2.0: Learning generative dexterous grasping in large-scale synthetic cluttered scenes. In *8th Annual Conference on Robot Learning*, 2024.
- [66] Ruijie Zheng, Dantong Niu, Yuqi Xie, Jing Wang, Mengda Xu, Yunfan Jiang, Fernando Castañeda, Fengyuan Hu, You Liang Tan, Letian Fu, et al. Egoscale: Scaling dexterous manipulation with diverse egocentric human data. *arXiv preprint arXiv:2602.16710*, 2026.

APPENDIX

A. Additional Ablation Results

We repeat the play pretraining ablations from the main paper on *Screw-Leg* to evaluate whether the observed design principles generalize beyond *Tight-Insertion*. We finetune each pretrained policy using the same sparse assembly rewards and training procedure as *Play2Perfect*. All curves report the success rate during assembly finetuning across three seeds.

Results. Fig. 4 shows that the conclusions from *Tight-Insertion* also hold on *Screw-Leg*. First, increasing **object diversity** improves both learning stability and final performance. Pretraining on 1000 objects reaches approximately 80% success, whereas the 10- and 100-object variants improve initially but remain unstable and finish substantially lower. This suggests that broader object diversity becomes particularly valuable when transferring to assembly tasks whose geometry differs significantly from the objects encountered during play. Second, the **training objective** must teach orientation control. The full 6D pose objective performs best, while Rotation-only also enables successful finetuning but learns more slowly and reaches slightly lower performance. Translation-only fails entirely, further demonstrating that grasping and transporting an object without learning in-hand reorientation does not provide a sufficient prior for assembly. Third, **trajectory diversity** is critical on this task. The policy pretrained with online random trajectories rapidly reaches approximately 80% success, while policies trained on fixed sets of 10 or 100 trajectories make little progress. Compared with the *Tight-Insertion* results, this larger gap indicates that broad coverage of object-pose transitions is especially important for transferring to substantially different manipulation sequences. Finally, precise play goals are necessary. With a **goal precision** of 1 cm, finetuning rapidly reaches high success, whereas the 5 cm and 10 cm variants fail to learn. Loose play objectives can therefore be solved without acquiring the fine object-pose control required to initialize precise, contact-rich assembly.

B. Simulation and Computational Resources

All policies are trained in Isaac Sim on a single NVIDIA RTX A6000 GPU. The physics simulation runs at 120 Hz, while the policy outputs actions at 60 Hz. Play pretraining uses 24,576 parallel simulation environments and is run for 7 days. Each downstream assembly policy is finetuned for 1 day using 12,228 parallel environments. We use fewer environments during finetuning because modeling contact-rich assembly interactions requires more GPU memory than the free-space manipulation used during play pretraining. For each wall-clock comparison, all methods within the same figure are run on the same GPU type with identical hyperparameters, environment counts, and training budgets.

C. Policy Architecture and RL Algorithm

We train the play policy using Split and Aggregate Policy Gradients (SAPG) [49], a population-based variant of PPO that improves exploration in massively parallel environments.

TABLE II: Play policy architecture and SAPG hyperparameters.

Parameter	Value
Actor network	LSTM[1024] + MLP[1024, 1024, 512, 512]
Critic network	MLP[1024, 1024, 512, 512]
Learning rate	1×10^{-4}
Minibatch size	98,304
SAPG block size	4,096
Entropy bonus scale	0.002
Discount factor γ	0.99
GAE parameter λ	0.95
PPO clip range	0.1

The actor uses an LSTM to integrate interaction history and infer unobserved object properties, followed by a multilayer perceptron that outputs the arm and hand actions. We use an asymmetric actor-critic architecture. While the actor receives only the observations available at deployment, the critic additionally receives noise-free and undelayed observations, palm and object velocities, reward signals, and stateful progress features such as the minimum goal distance reached and whether the object has been lifted. This privileged information is used only during training. The same RL algorithm and hyperparameters are used for both pretraining and finetuning.

D. Play Pretraining Details

Pretraining Environment and Procedural Objects. At the start of each episode, a procedurally generated object is placed at a random pose on the table. The robot must grasp and lift the object, then manipulate it through a sequence of randomly sampled 6D goal poses. Each object is formed by rigidly combining two cuboid or capsule primitives. The primary component defines the graspable region, and its length and cross-sectional dimensions are sampled from [5, 30] cm. A secondary component is attached near one end, with length sampled from [1, 15] cm and cross-sectional dimensions from [0.5, 12] cm. We independently randomize the component densities, sampling the primary component from [300, 600] kg/m³ and the secondary component from [300, 2000] kg/m³. This produces broad variation in geometry, mass, center of mass, and inertia.

Episode Initialization and Goal Sampling. At the start of each episode, the robot is initialized around a default joint configuration with uniform noise of ± 0.1 rad. The object is placed above the center of the table with its position randomized within ± 10 cm and its orientation sampled randomly. The first goal is sampled within the robot’s reachable workspace, with position relative to the table center sampled as $x \in [-0.35, 0.35]$ m, $y \in [-0.1, 0.2]$ m, and $z \in [0.15, 0.52]$ m. After reaching a goal, the next goal is sampled relative to it with a translation of up to 0.1 m and a rotation of up to 90°. This produces smooth but diverse goal trajectories that require repeated in-hand reorientation. Each episode lasts at most 600 control steps (10 s). It terminates early if the object falls below the table, the hand moves more than 1.5 m from

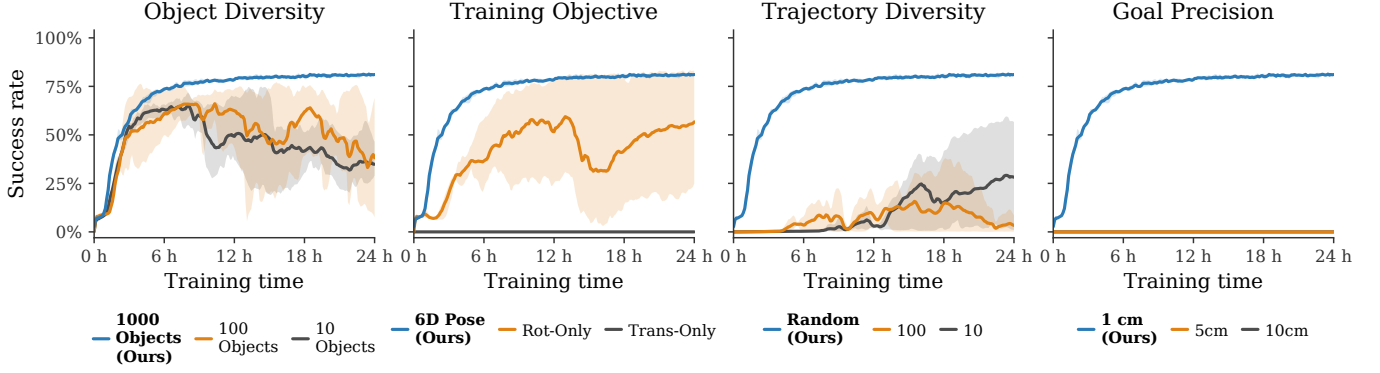


Fig. 4: **Additional pretraining ablations on Screw-Leg.** We repeat the four play pretraining ablations from the main paper on a second downstream assembly task. The results reinforce the same overall conclusion: pretraining transfers best when it teaches precise 6D in-hand object control across diverse objects and goal trajectories. On this task, weaker object-diversity, trajectory-diversity, and goal-precision variants degrade more sharply than on Tight-Insertion.

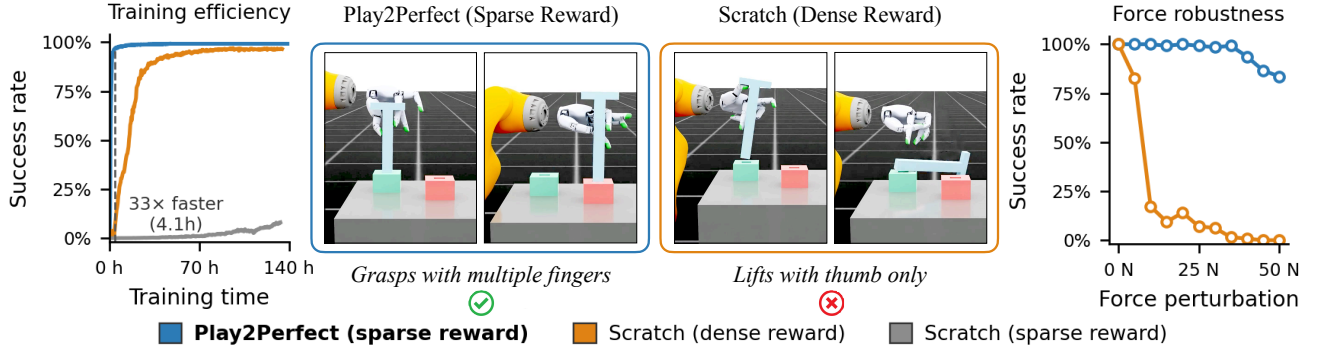


Fig. 5: **Dexterous Play Pretraining Induces Robust Assembly Strategies.** After simplifying the initialization with an easy-grasp fixture, Scratch (dense reward) can learn the task, but it relies on a brittle strategy that balances the object rather than robustly grasping it. This shortcut fails sharply under external force perturbations. In contrast, *Play2Perfect* learns a more stable grasping and recovery strategy, maintaining high success across perturbation magnitudes.

the object, the contact force measured at the table exceeds 100 N, or the maximum number of goal successes is reached.

Keypoint-Based Pose Representation. We represent each 6D object pose using four keypoints defined in the local object frame. Given dimensions $\mathbf{s} = [s_x, s_y, s_z]$, the keypoints are placed at

$$\mathcal{K}(\mathbf{s}) = \left\{ \begin{bmatrix} s_x/2 \\ s_y/2 \\ s_z/2 \end{bmatrix}, \begin{bmatrix} s_x/2 \\ -s_y/2 \\ -s_z/2 \end{bmatrix}, \begin{bmatrix} -s_x/2 \\ s_y/2 \\ -s_z/2 \end{bmatrix}, \begin{bmatrix} -s_x/2 \\ -s_y/2 \\ s_z/2 \end{bmatrix} \right\}. \quad (1)$$

For a pose $\mathbf{o} = (R_o, \mathbf{t}_o)$, each keypoint is transformed into the world frame as

$$\mathbf{o}_i = R_o \mathbf{k}_i + \mathbf{t}_o. \quad (2)$$

We define the distance between current and goal poses as

$$d(\mathbf{o}, \mathbf{g}) = \max_i \|\mathbf{o}_i - \mathbf{g}_i\|_2. \quad (3)$$

This provides a single metric that jointly captures translation and rotation error. For policy observations, we define the keypoints using the dimensions of the object’s primary component. For reward computation, we instead use fixed dimensions

$\mathbf{s}^{\text{rew}} = [0.14, 0.03, 0.03]$ m, ensuring a consistent trade-off between translational and rotational errors across objects.

Policy Observations and Action Space. The policy receives a 140-dimensional observation containing robot proprioception, the current object pose, the goal pose, and an object geometry descriptor. Robot proprioception includes the 29 joint positions and velocities, previous joint-position targets, palm pose, and the positions of the five fingertips relative to the palm. We represent the current and goal object poses using four keypoints defined by the dimensions of the object’s primary component. The policy observes the object orientation, the current keypoints relative to the palm, the keypoint displacements from the current pose to the goal, and the three object dimensions. Using relative keypoint representations reduces dependence on absolute workspace coordinates while conditioning the policy on object geometry. The policy outputs 29 joint-position commands for the 7-DoF arm and 22-DoF hand. Arm actions are represented as delta joint-position targets, while hand actions are represented as absolute joint-position targets. All commands are clipped to the corresponding joint limits and smoothed using an

exponential moving average with coefficient $\alpha = 0.1$.

Play Reward and Success Criterion. The play reward combines smoothness, grasping, and goal-reaching terms:

$$r = r_{\text{smooth}} + r_{\text{grasp}} + \mathbb{I}_{\text{grasped}} r_{\text{goal}}. \quad (4)$$

We encourage smooth control by penalizing arm and hand joint velocities:

$$r_{\text{smooth}} = -\lambda_{\text{arm}} \|\dot{\mathbf{q}}^{\text{arm}}\|_1 - \lambda_{\text{hand}} \|\dot{\mathbf{q}}^{\text{hand}}\|_1. \quad (5)$$

The grasp reward first encourages the fingertips to approach the object and then lift it from the table:

$$r_{\text{grasp}} = r_{\text{approach}} + (1 - \mathbb{I}_{\text{grasped}}) r_{\text{lift}}, \quad (6)$$

$$r_{\text{approach}} = \lambda_{\text{approach}} \max(\bar{d}_{\text{ft}}^* - \bar{d}_{\text{ft}}, 0), \quad (7)$$

$$r_{\text{lift}} = \lambda_{\text{lift}} \max(z - z_{\text{init}}, 0) + B_{\text{lifted}} \mathbb{I}[z \geq z_{\text{lifted}}], \quad (8)$$

where $\bar{d}_{\text{ft}}$ is the mean fingertip-to-object distance and $\bar{d}_{\text{ft}}^*$ is the smallest distance reached so far in the episode. We set $\mathbb{I}_{\text{grasped}} = 1$ once the object has been lifted by 10 cm. After the object is grasped, the policy is rewarded for making progress toward the current 6D goal:

$$r_{\text{goal}} = \lambda_{\text{goal}} \max(d^* - d(o_t, g_t), 0) + B_{\text{succ}} \mathbb{I}[d(o_t, g_t) < \epsilon], \quad (9)$$

where d^* is the smallest goal distance reached since the current goal was sampled. Reaching a goal within $\epsilon = 1$ cm yields a sparse success bonus, after which a new goal is sampled. For reward computation, we use the keypoint-based pose distance defined above with fixed dimensions $\mathbf{s}^{\text{rew}} = [0.14, 0.03, 0.03]$ m. This provides a consistent trade-off between translation and rotation error across objects. We use $\lambda_{\text{arm}} = 0.03$, $\lambda_{\text{hand}} = 0.003$, $\lambda_{\text{approach}} = 50$, $\lambda_{\text{lift}} = 20$, $\lambda_{\text{goal}} = 200$, $B_{\text{lifted}} = 300$, and $B_{\text{succ}} = 1000$.

TABLE III: **Domain randomization used during play pre-training.**

Parameter	Randomization
Current/goal pose noise (translation)	1 cm
Current/goal pose noise (rotation)	5°
Object-pose delay	0–10 control steps
Action and proprioception delay	0–3 control steps
Joint-velocity observation noise	$\sigma = 0.1$ rad/s
Object-dimension scale	90%–110%
Table-height variation	± 1 cm
External force perturbation	20.0 N
External torque perturbation	2.0 N m

Domain Randomization. We apply domain randomization to observations, action execution, object geometry, and environment dynamics to improve sim-to-real transfer. We perturb the observed current and goal object poses, introduce latency in object-pose estimates, actions, and proprioceptive observations, and add noise to joint-velocity observations. We add noise to the object geometry descriptor and vary table height, and apply random forces and torques to the manipulated object, encouraging the policy to learn stable grasps and recovery behaviors. Table III lists the values.

E. Pretraining Ablation Implementations

We ablate four components of play pretraining: object diversity, training objective, trajectory diversity, and goal precision. Unless otherwise stated, all variants use the same environment, policy architecture, reward coefficients, domain randomization, training budget, and downstream assembly-finetuning procedure as the default *Play2Perfect* policy. Each ablation changes only the pretraining factor under study.

Object Diversity. We vary the number of procedurally generated objects used during pretraining among 10, 100, and 1000, with 1000 objects used by default. Each object set is sampled from the same procedural distribution described in Sec. D, including the same geometry and density ranges. During pretraining, each environment samples an object from the corresponding fixed object set. All goal-sampling and optimization settings remain unchanged.

Training Objective. We compare the default 6D pose-reaching objective against Translation-only and Rotation-only variants. The default objective requires jointly matching the target translation and orientation using the keypoint-based pose distance. Translation-only uses the same sampled goal sequences, but computes goal success using only the Euclidean distance between the current and target object positions. The target orientation is ignored. Rotation-only first requires the policy to grasp, lift, and move the object to a randomly sampled 6D pose. Subsequent goals keep this target position fixed and vary only the target orientation, requiring repeated in-hand reorientation without translational goal changes.

Trajectory Diversity. The default policy samples every goal trajectory online during pretraining. To study the effect of trajectory diversity, we instead construct fixed banks containing either 10 or 100 goal trajectories. Each trajectory follows the same sampling distribution as the default setting, including the same workspace bounds and maximum translation and rotation between consecutive goals. During pretraining, episodes repeatedly sample from the corresponding fixed trajectory bank rather than generating new trajectories online.

Goal Precision. We vary the success tolerance used to determine whether a play goal has been reached. The default policy uses $\epsilon = 1$ cm, while the ablations use $\epsilon = 5$ cm or $\epsilon = 10$ cm. Only the success threshold is changed. The sampled goals, pose representation, reward coefficients, and all other training settings remain identical.

F. Assembly Finetuning Details

Environment Construction from CAD. For each assembly step, we import the manipulated part and fixture CAD models into Isaac Sim as rigid bodies. Most geometry is represented using convex decomposition for efficient simulation. However, convex approximations can distort narrow holes and mating interfaces, changing the effective clearance and contact dynamics. We therefore represent only the contact-critical hole and insertion components using signed distance fields (SDFs) at resolution 256. This targeted hybrid representation provides detailed collision geometry where precision matters while

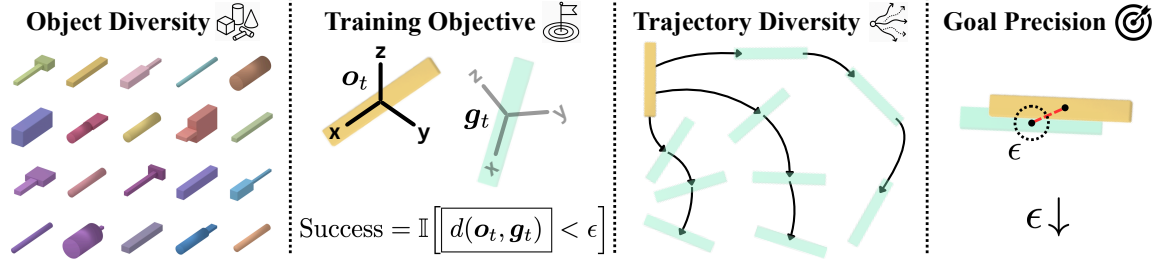


Fig. 6: **What matters in dexterous *play* pretraining?** We study the key factors that shape the learned manipulation prior. Our design emphasizes in-hand manipulation with fingers across diverse objects and trajectories, with 6D pose-reaching objectives and precise goal tolerances.

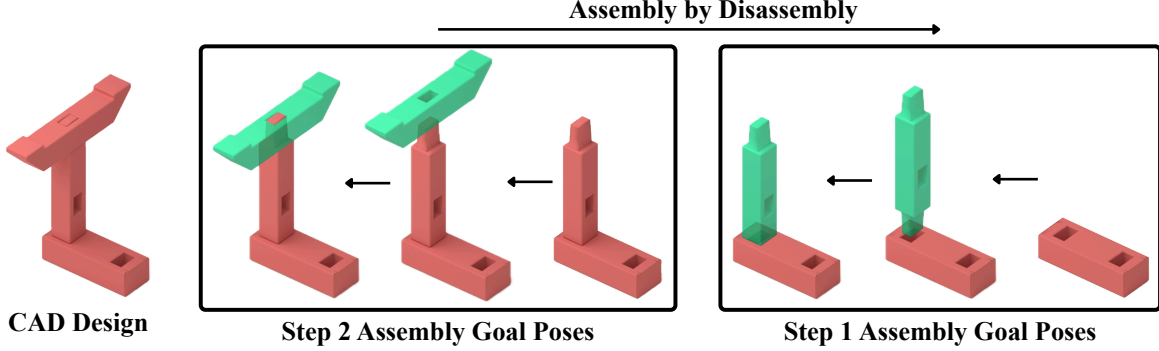


Fig. 7: **Assembly-by-Disassembly.** Given a completed CAD assembly, we generate assembly steps by sequentially removing parts and reversing the disassembly sequence. Each step defines a sparse goal sequence: the final assembled pose and intermediate contact goals, e.g., pre-insert pose.

avoiding the memory and computational cost of applying high-resolution SDFs to the entire assembly in simulation.

Episode Initialization and Reset Distribution. At the beginning of each episode, the manipulated part is spawned above the table with its planar position sampled independently as $x, y \in [-0.1, 0.1]$ m and its orientation sampled uniformly at random. The part is then dropped onto the table, producing diverse stable initial poses. The fixture is placed flat at a random location on the table.

CAD-Derived Assembly Goal Sequences. For each assembly step, the CAD design specifies the desired transform T_p^f of the manipulated part p in the fixture frame f . Given the current fixture pose $f_t \in SE(3)$, we compute the final world-frame goal as

$$g_M = f_t T_p^f. \quad (10)$$

This makes the goal invariant to the randomized fixture placement. For contact-rich tasks, we augment the final assembled pose with a small sequence of sparse intermediate goals obtained by reversing the corresponding disassembly motion. For insertion tasks, this includes an aligned pre-insertion pose immediately before contact, followed by the final inserted pose. For screwing, we generate successive goals along the thread with 90° rotational offsets. The policy advances to the next goal once the current pose is reached within the specified tolerance, and the final goal corresponds to successful completion of the assembly step.

Assembly Reward, Goal Progression, and Success. Assembly finetuning uses a sparse task reward derived only from the CAD-defined goal sequence. We remove the grasping, lifting, and dense pose-progress rewards used during play pretraining. The finetuning reward contains only a smoothness regularizer and sparse bonuses for reaching the assembly goals:

$$r_t = r_{\text{smooth}} + r_{\text{goal}}. \quad (11)$$

We retain the same smoothness regularizer as during pretraining:

$$r_{\text{smooth}} = -\lambda_{\text{arm}} \|\dot{\mathbf{q}}^{\text{arm}}\|_1 - \lambda_{\text{hand}} \|\dot{\mathbf{q}}^{\text{hand}}\|_1. \quad (12)$$

Let g_m denote the active CAD-derived goal. The policy receives no task-dependent reward as it approaches the goal. It receives a sparse success bonus only once the part enters the goal tolerance:

$$r_{\text{goal}} = B_{\text{succ}} \mathbb{I}[d(o_t, g_m) < \epsilon] + r_{\text{retract}}, \quad (13)$$

where $d(\cdot, \cdot)$ is the keypoint-based 6D pose distance defined in Sec. D. When an intermediate goal is reached within $\epsilon = 1$ cm, the environment advances to the next goal in the sequence. Reaching the final goal marks the assembly itself as successful. For the final goal, we additionally provide a sparse retraction bonus once the assembled part remains at its goal and the

robot moves its palm more than 0.2 m away:

$$r_{\text{retract}} = B_{\text{retract}} \mathbb{I} \left[d(o_t, g_M) < \epsilon \wedge \left\| \mathbf{p}_t^{\text{palm}} - \mathbf{p}_t^{\text{obj}} \right\|_2 > 0.2 \text{ m} \right]. \quad (14)$$

This encourages the robot to release and retract after completing the assembly, rather than relying on continued hand contact to hold the part at its final pose. We use $B_{\text{succ}} = B_{\text{retract}} = 1000$. Thus, apart from action smoothness, the environment provides feedback only when the policy reaches a CAD-derived goal or completes the final retraction. There are no explicit rewards for approaching, grasping, lifting, alignment, contact, or reducing pose error. These behaviors must instead be retained from the pretrained play policy and adapted through sparse-reward finetuning.

Domain Randomization During Finetuning. We apply domain randomization during assembly finetuning to improve robustness to perception errors, control latency, and contact-dynamics mismatch. We perturb the observed current part pose and CAD-derived goal pose with independent translational and rotational noise. Goal-pose noise models errors in the estimated fixture pose used to compute the assembly goals. We also randomize the physical fixture yaw, object-pose latency, action and proprioception latency, joint-velocity observations, observed part dimensions, and table height. Finally, random external forces and torques are applied to the manipulated part to encourage stable grasping and recovery during contact-rich interactions. Table IV summarizes the parameters.

TABLE IV: Domain randomization used during assembly finetuning.

Parameter	Randomization
Current part-pose noise (translation)	1 cm
Current part-pose noise (rotation)	5°
Goal-pose noise (translation)	2 mm
Goal-pose noise (rotation)	1°
Fixture yaw	[−10°, 10°]
Object-pose delay	0–10 control steps
Action and proprioception delay	0–3 control steps
Joint-velocity observation noise	$\sigma = 0.1 \text{ rad/s}$
Part-dimension noise scale	90%–110%
Table-height variation	±1 cm
External force perturbation	20.0 N
External torque perturbation	2.0 N m

G. Inference Time Pipeline

At deployment, we reuse the part and fixture CAD models from training in three ways. (1) Part pose estimation: FoundationPose [59] tracks the manipulated part pose online and provides the current object pose to the policy. (2) Goal pose: the fixture pose is estimated once at the beginning of each rollout. Since the fixture remains fixed during execution, the desired part goal poses are then computed from the estimated fixture pose and the CAD-specified assembly transforms. (3) Grasp bounding box: we define a grasp bounding box on the part CAD model that specifies the region where the robot should grasp the object.

The finetuned policy runs as a closed-loop controller at 60 Hz. At each control step, it receives robot proprioception, the current part pose, the desired goal pose, and the grasp bounding box. The policy outputs joint position targets for the 7-DoF arm and 22-DoF dexterous hand, which are executed by the robot’s low-level joint position controllers. The part pose tracker runs at 30 Hz, and the controller reuses the most recent pose estimate between tracking updates. We do not use an additional scripted insertion, screwing, or recovery controller at deployment. Contact-rich behaviors such as local search, corrective motions, regrasping, and in-hand spinning are produced by the learned policy.

For each assembly task, the policy follows a short sequence of sparse goal poses $\mathcal{G} = g_1, \dots, g_M$ derived from the CAD assembly. At the beginning of a rollout, the active goal is initialized to the first goal g_1 . During execution, we compute the pose distance between the current part pose o_t and the active goal g_m . When $d_{\text{pose}}(o_t, g_m) < \epsilon_{\text{goal}}$, the controller advances to the next goal, g_{m+1} . This process repeats until the final goal g_M is reached, at which point the assembly is considered complete. For example, insertion tasks use intermediate pre-insertion goals before the final inserted pose, while screwing tasks use goal poses along the thread to induce axial rotation.

Fig. 9 visualizes the policy observations during a real-world Screw-Leg rollout. The policy receives the estimated current part pose and the active sparse goal pose at each timestep. As the rollout progresses, the active goal advances through the CAD-derived goal sequence.

H. Real-World Experiment Additional Analysis

Qualitative behavior. Across real-world tasks, *Play2Perfect* exhibits closed-loop recovery behaviors that are difficult to obtain from a single open-loop assembly motion. During insertion, the policy often searches locally near the hole, makes small corrective motions under contact, and commits once the part is better aligned. When the part becomes misaligned, the policy can recover with additional in-hand reorientations. If the part is dropped, the policy immediately regrasps and continues, provided the object remains within the workspace. In screwing tasks, the fingers spin the leg directly within the hand, enabling axial rotation while maintaining a stable grasp. With a parallel-jaw gripper, comparable reorientation would typically require slower extrinsic manipulation, such as placing and regrasping the part [32], and screwing would require rotating the entire arm around the screw axis rather than spinning the part within the fingers [63].

Fig. 10 shows a representative real-world Tight-Insertion rollout. After finetuning, the policy does not move directly to the final pose. Instead, it approaches the hole with a tilted insertion strategy, makes contact near the fixture, and uses small closed-loop corrective motions to search locally before committing to insertion. This behavior becomes especially important at tighter clearances, where small pose errors are enough to block direct insertion.

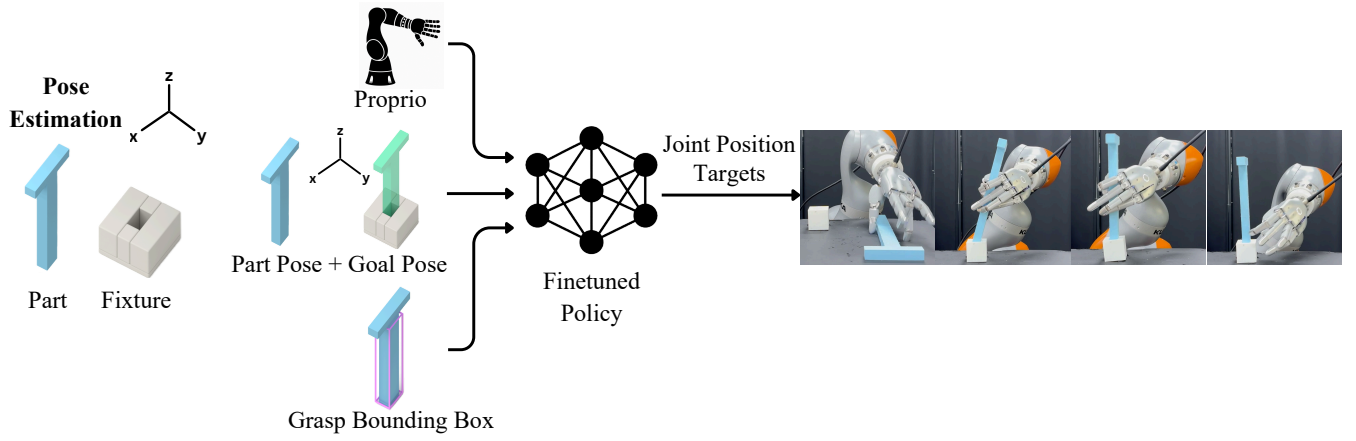


Fig. 8: **Inference-Time Pipeline.** At deployment, CAD models are reused to estimate the current part pose, compute CAD-derived goal poses relative to the fixture, and define the grasp bounding box. The finetuned policy takes robot proprioception, part pose, goal pose, and the grasp bounding box as input, and outputs joint position targets for the arm and dexterous hand.

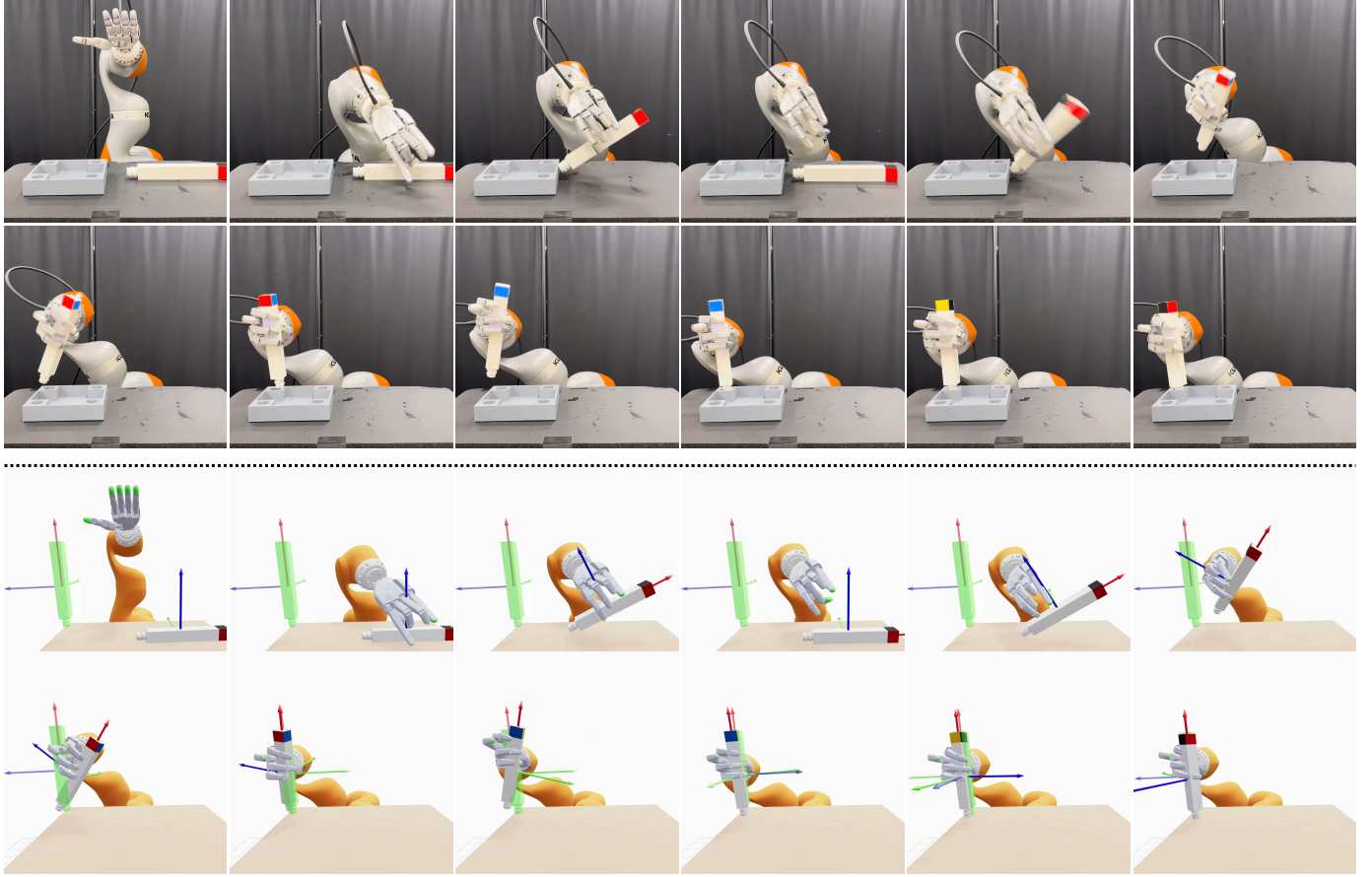


Fig. 9: **Policy Observations During Real-World Deployment.** (Top) A representative real-world Screw-Leg rollout, with (Bottom) the corresponding policy observations. The translucent green object and translucent axes denote the active sparse goal pose, while the opaque object and full-opacity axes denote the estimated current part pose observed by the policy. As the rollout progresses, the active goal advances along the screw axis with successive rotational offsets, and the policy tracks these goals by reorienting the leg in-hand. During fast motions, the visualized current pose can lag the real object due to object-pose tracking latency.

Fig. 11 shows that the policy remains closed-loop after early failures. Even after an initial failed grasp or drop, the policy

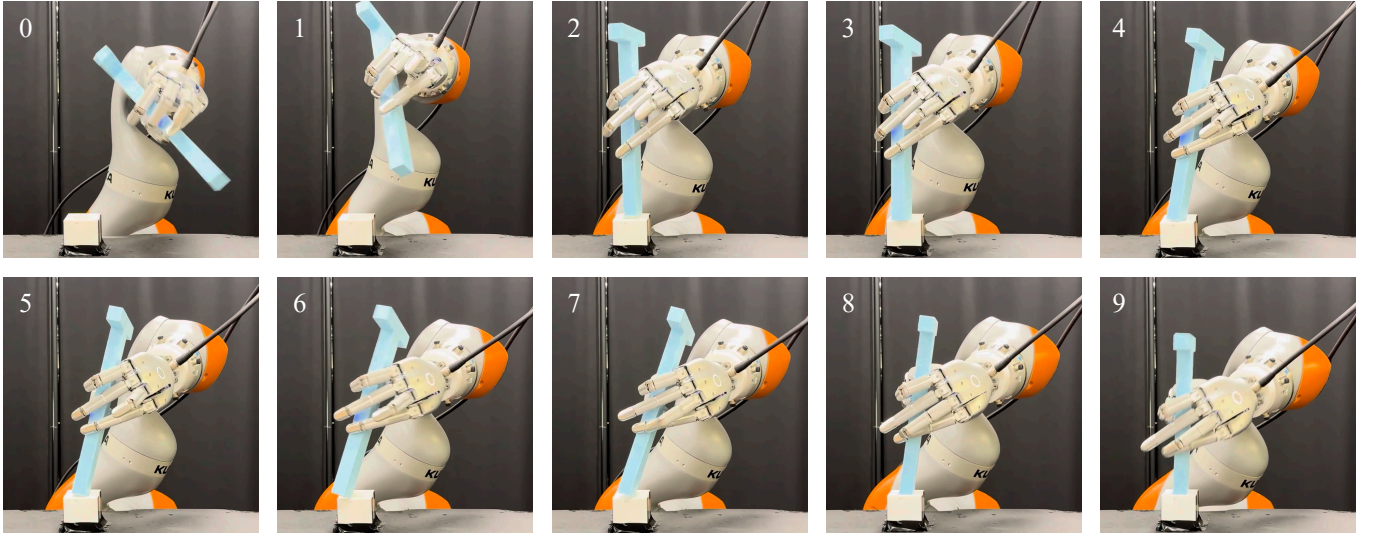


Fig. 10: **Tilted Insertion and Local Search.** A representative real-world *Tight-Insertion* rollout. The policy approaches the hole with a tilted insertion strategy, makes contact near the fixture, and performs a local search with small corrective motions before committing to insertion.

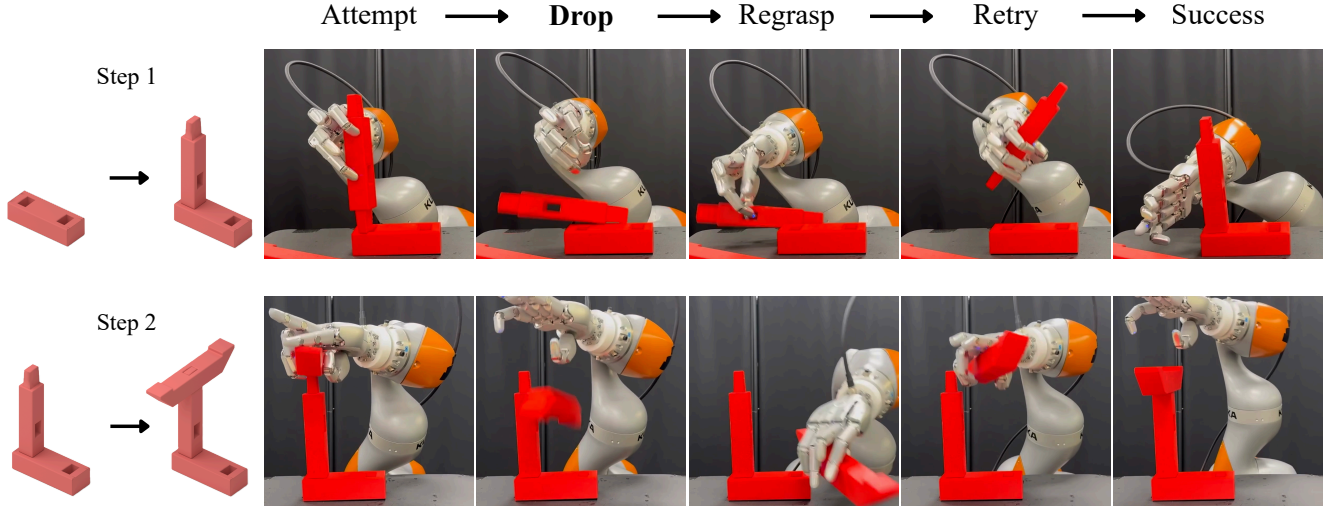


Fig. 11: **Recovery After Drops.** Representative real-world *Assemble-Beam* rollouts for both assembly steps. After dropping the part, the policy continues acting closed-loop, regrasps the object, retries the assembly motion, and completes the task without a scripted recovery controller.

continues acting from the new state, regrasps the part, retries the assembly motion, and completes the task without a scripted recovery controller.

Failure modes. Real-world failures arise from both perception and contact dynamics. Perception remains a major failure mode even on larger parts: fast part motion, hand-object occlusion, and visually similar objects can cause the pose estimator to lose track of the manipulated part. Screwing is especially challenging because the policy must track object rotation, while the rectangular leg has approximate 90° rotational symmetries that can confuse pose estimation and cause the policy to rotate in the wrong direction. We add colored tape to each side of the rectangular leg to make

the orientations visually distinguishable, but pose estimation remains imperfect during fast rotations and occlusions. Control failures typically occur during the final contact-rich phase, when the policy repeatedly attempts insertion but fails to align with the hole, or contacts the fixture in ways that differ from simulation. In simulation, fixtures are rigid and immovable, while in the real setup, they are taped to a foam tabletop for safety and can move or comply under contact. This behavior is never observed in simulation and can cause the policy to struggle when its corrective motions no longer produce the expected relative motion between the part and fixture. We also observe some failed in-hand reorientations and drops during rotation before insertion, requiring the policy to

regrasp. In contrast, grasp acquisition is highly reliable: real-world failures rarely arise from missed grasps except when perception fails catastrophically.

I. Assembly Finetuning Is Necessary for Tight Insertion

Setup. We evaluate whether *play* pretraining alone is sufficient for precise assembly, or whether task-specific RL finetuning is necessary. We compare *Play2Perfect* against *Play-only*, the pretrained policy without assembly finetuning. We evaluate both policies on *Tight-Insertion* across fixture holes with varying contact clearances. Clearance is the difference between the hole and peg cross-sectional dimensions: for example, our peg has a $30\text{mm} \times 20\text{mm}$ cross-section, so a 1mm clearance corresponds to a $31\text{mm} \times 21\text{mm}$ hole. We finetune *Play2Perfect* on 5–0.5 mm clearances, then evaluate both policies from loose 40 mm to tight 0.2 mm clearance.

Results. Fig. 12 shows that *Play-only* solves only the loosest insertion settings. In simulation, it reaches 75% success at 40 mm clearance but drops to nearly 0% by 4 mm. In contrast, *Play2Perfect* maintains high success, achieving 95% at 4 mm, 92% at 1 mm, and 80% at 0.2 mm, which is tighter than the training distribution. This shows that *play* pretraining provides useful grasping and reorientation behaviors, but RL finetuning is needed to turn this prior into a precise, contact-rich assembly policy. The real-world results show the same trend. At 10 mm clearance, *Play2Perfect* achieves 100% success compared to 60% for *Play-only*. At 2 mm clearance, *Play2Perfect* achieves 90% success while *Play-only* drops to 20%. At 0.5 mm clearance, *Play2Perfect* succeeds 60% of the time, whereas *Play-only* fails completely. Qualitatively, *Play-only* tends to move directly toward the goal pose and treats contact as a disturbance. In contrast, *Play2Perfect* learns to search locally near the hole, make corrective motions under contact, and commit to insertion once the part is aligned.

J. Related Work

Manipulation with Multi-Fingered Robots. Prior work falls into two categories: imitation learning (IL) and reinforcement learning (RL). IL relies on collecting high-quality demonstrations, which can be obtained through human hand motion retargeting with motion-capture gloves [43, 58], VR devices [15, 21, 28, 6], camera inputs [39, 18, 50], or exoskeleton systems [53, 60, 16]. However, collecting demonstrations for contact-rich tasks remains difficult due to the embodiment gap between the human operator and the robot, as well as the lack of tactile feedback [10, 45, 5, 37]. Sim-to-real RL offers a promising alternative, with recent progress on dexterous skills such as grasping [1, 62, 30, 48, 47] and in-hand reorientation [11, 12, 29]. Yet, these skills are largely performed in free space. Extending dexterous RL to contact-rich tasks relies on dense reward functions [27, 13], accurate human hand motion references [26, 31, 34], or warm-starting from teleoperation [46, 9]. Closest to our work are methods that train task-agnostic *play* controllers across diverse objects [64, 23, 25]. When combined with teleoperation [64] or a human demonstration at test time [23], these controllers

can generalize to unseen objects and tasks. However, they are deployed zero-shot and still struggle with precise, contact-rich tasks such as assembly. Instead, we view *play* as dexterous pretraining: a general prior that is specialized to assembly.

Precise and Contact-Rich Assembly. Most progress in contact-rich assembly has come from structuring the problem through task-specific hardware or environments. Prior work designs specialized gripper attachments or tools to simplify manipulation [17, 44], or uses fixtures to reduce uncertainty in grasping, alignment, and insertion [42, 32]. While effective, these approaches require task-specific setup for each assembly problem. Learning-based assembly methods often rely on task-specific structure as well, including dense reward functions [51, 52], scripted multi-stage controllers [55], or carefully designed curricula [22]. Other methods address the exploration problem with dense reset distributions [63] or teleoperation demonstrations [3, 4], but these still require task insight or task-specific data. In contrast, we pretrain a general dexterous *play* prior without demos or knowledge of the downstream assembly task, and specialize it with sparse-reward RL.

Pretraining for Dexterous Manipulation. There is growing interest in pretraining dexterous priors to solve challenging manipulation tasks. For example, Vision Language Action (VLA) models [38, 35, 8] train on large datasets consisting of diverse tasks [36, 24]. However, these datasets are largely concentrated on parallel jaw gripper robots. Human videos on the internet are another promising option to acquire priors for dexterous hands [66, 61, 40, 53]. However, human videos do not contain contact information, limiting their use to simple pick and place tasks and often requiring large amounts of in-domain robot finetuning data. Learning from *play* is a promising approach to learn priors from task-agnostic data. Prior works still require teleoperation to collect robot trajectories or active human hand data collection [57, 33]. Compared to these works, we train *play* priors by learning to manipulate diverse objects via RL without requiring any demonstrations.

K. Limitations

Our system learns short-horizon assembly skills rather than a complete autonomous assembly pipeline. Task sequencing, active-part selection, and goal poses are specified externally, and policies are finetuned per task or benchmark family. Future work could combine these skills with sequencing, scene memory, recovery, and broader multi-task finetuning. Real-world deployment also depends on object-pose estimates, which can fail under occlusion or fast motion. Finally, beyond the goal pose, the policy does not directly observe the fixture or surrounding geometry. Incorporating visual or tactile observations could address these perception and scene-awareness limitations.

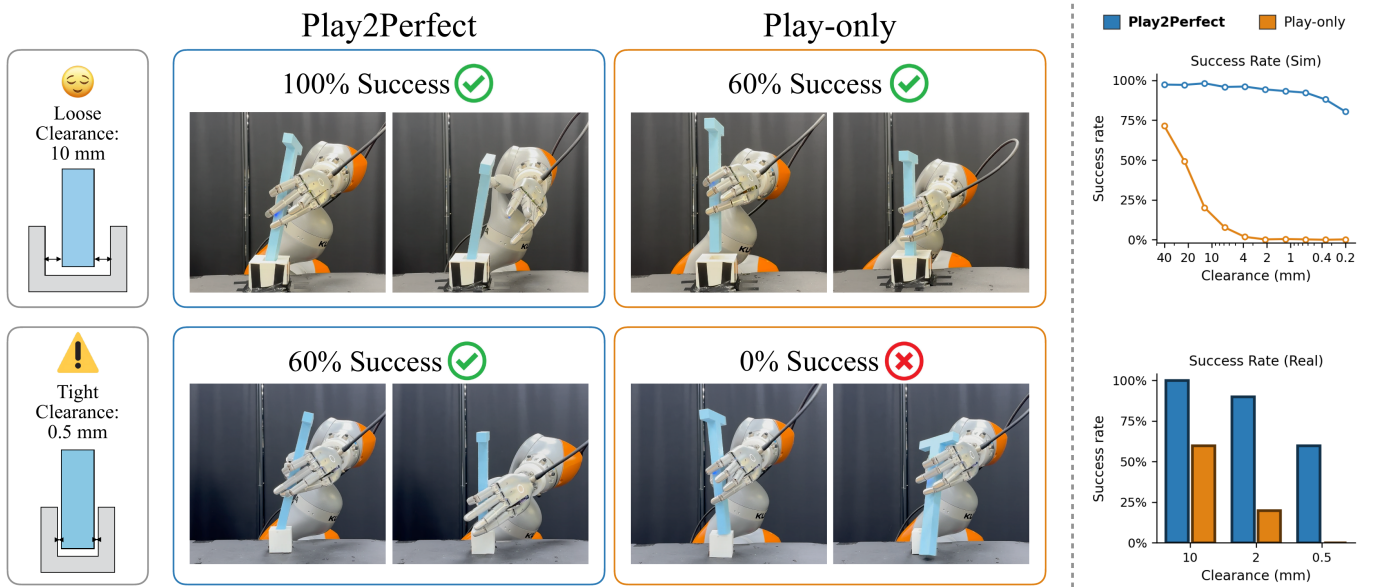


Fig. 12: **Assembly Finetuning Enables Tight Insertion.** We compare *Play2Perfect* against a frozen *Play-only* policy on insertion tasks with varying contact clearance. (Left) Both policies succeed at loose clearance, but only *Play2Perfect* succeeds at tight clearance. (Top right) Simulation sweeps show that *Play2Perfect* remains robust as clearance decreases, while *Play-only* rapidly degrades. (Bottom right) Real-world success rates show the same trend across different clearances.