SimToolReal: An Object-Centric Policy for Zero-Shot Dexterous Tool Use

anon-simtoolreal.github.io

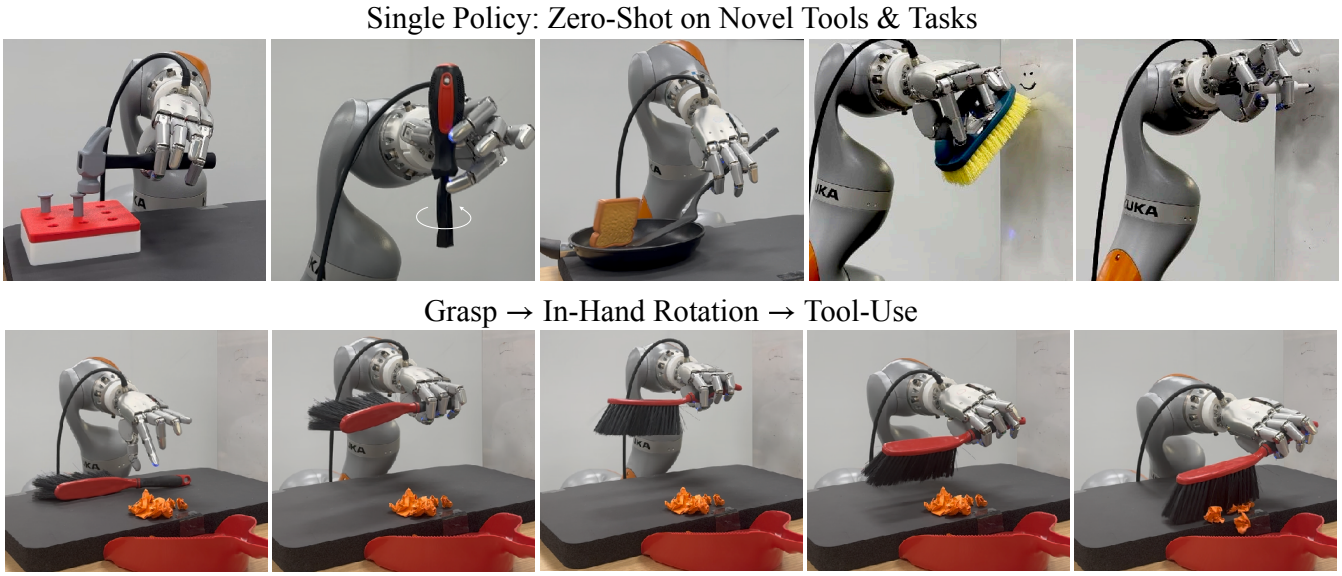

Fig. 1: ***SimToolReal*** is a framework for training a single general-purpose, object-centric RL policy in simulation and transferring it to real-world tool use. (Top) Zero-shot deployment on novel real tools and tasks, spanning thin markers to thick hammers. (Bottom) Tool use typically involves grasping objects from flat surfaces, reorienting them in-hand, and performing the task.

***Abstract*— The ability to manipulate tools significantly expands the set of tasks a robot can perform. Yet, tool manipulation represents a challenging class of dexterity, requiring grasping thin objects, in-hand object rotations, and forceful interactions. Since collecting teleoperation data for these behaviors is challenging, sim-to-real reinforcement learning (RL) is a promising alternative. However, prior approaches typically require substantial engineering effort to model objects and tune reward functions for each task. In this work, we propose *SimToolReal*, taking a step towards generalizing sim-to-real RL policies for tool manipulation. Instead of focusing on a single object and task, we procedurally generate a large variety of tool-like object primitives in simulation and train a single RL policy with the universal goal of manipulating each object to random goal poses. This approach enables *SimToolReal* to perform general dexterous tool manipulation at test-time without any object or task-specific training. We demonstrate that *SimToolReal* outperforms prior retargeting and fixed-grasp methods by 37% while matching the performance of specialist RL policies trained on specific target objects and tasks. Finally, we show that *SimToolReal* generalizes across a diverse set of everyday tools, achieving strong zero-shot performance over 120 real-world rollouts spanning 24 tasks, 12 object instances, and 6 tool categories.**

I. Introduction

From household chores to industrial work, tool use expands a robot's capacity to act on its environment. However, tool use represents a particularly challenging class of dexterous manipulation tasks. Success requires grasping tools lying flat on a surface, reorienting them into functional poses, and maintaining control during potentially forceful interactions with the environment (see Fig. 1). Consider hammering a nail: the robot must grasp the hammer by its thin handle, rotate it in-hand into a striking configuration, and deliver impact without losing the grasp.

Imitation learning from teleoperated demonstrations is a common paradigm for learning manipulation policies [1], [2], but teleoperation is a poor fit for collecting high-quality dexterous tool-use data [3]. Human and robot hands differ in kinematics and actuation, creating a human-to-robot correspondence gap [4], [5] that makes precise dexterous control unintuitive for the operator [6]. At the same time, operators receive limited to no force and tactile feedback, preventing reliable regulation of contacts during the task [7], [8], [9], [10].

Sim-to-real reinforcement learning (RL) is a promising alternative that has successfully demonstrated agile, dynamic, and dexterous behaviors in the real world [11], [12]. Although effective for specific skills, extending its success to diverse tools and tasks is bottlenecked by substantial per-object simulation setup and task-specific reward engineering [13]. Consequently, existing approaches are limited to sub-problems of dexterous tool-use, such as grasping [14], [15], [16], in-hand object reorientation [17], [18], [19], or spinning objects [20], [21].

Our goal is to learn a dexterous manipulation policy that can be applied zero-shot to novel tools and tasks at test-

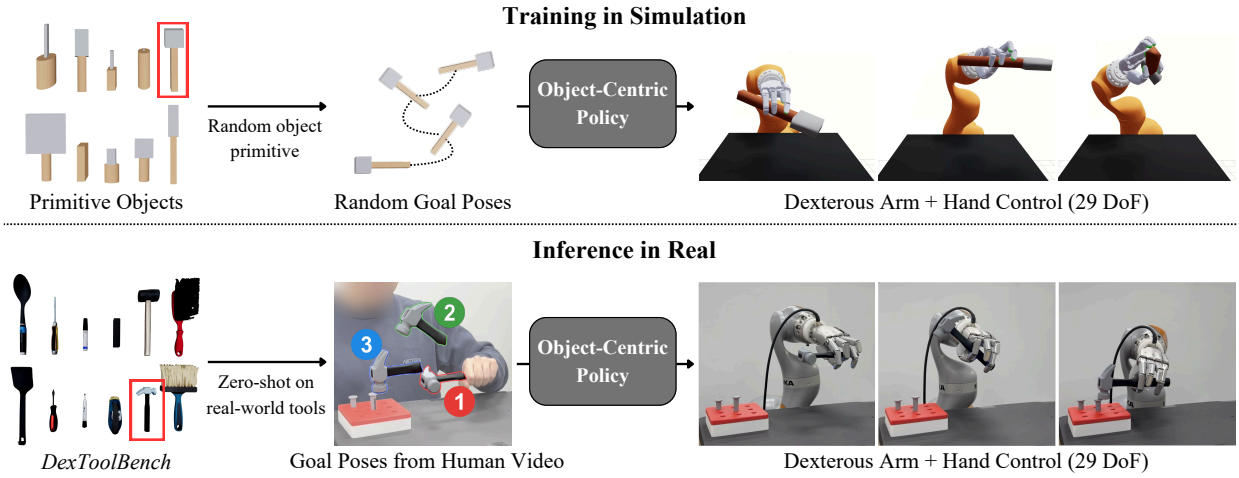

Fig. 2: **Overview of *SimToolReal*.** (Top) Training in Simulation: We train a goal-conditioned RL policy in simulation that manipulates a wide variety of procedurally-generated objects to randomly sampled goal poses. (Bottom) Inference in Real: We deploy this policy zero-shot on real-world tools from *DexToolBench*, following tool trajectories from human videos.

time. Our key insight is to view dexterous tool use from an *object-centric* lens, framing a task as moving a tool through a sequence of goal poses. Thus, we focus on training a single goal-conditioned policy that manipulates diverse objects to random goal poses. In simulation, we procedurally generate a large set of primitive objects and train one unified goal-conditioned policy. At test time, we compose this policy for zero-shot tool use by sequentially conditioning it on a goal pose trajectory extracted from a human video [22], [23]. This provides dense guidance from the initial grasp, through in-hand reorientation into a functional configuration, and into the ensuing tool-use motion. Our approach avoids per-task reward shaping while allowing generalization to novel tools and trajectories at test-time. Intuitively, mastering the ability of manipulating objects to any random pose induces the core skills required for tool use: establishing an initial grasp, in-hand object reorientation, and maintaining stable contact.

We instantiate this approach in *SimToolReal*, a framework for training a single general-purpose RL policy in simulation and deploying it for real-world tool use (Fig. 2). To enable generalization to novel objects, the policy is conditioned on the tool's current 6D pose and a coarse 3D bounding box over its graspable region (e.g., the handle of a hammer). This abstraction enables zero-shot transfer to the real-world by effectively bypassing the sim-to-real visual gap. At deployment, we recover this representation on real tools using a perception pipeline based on vision foundation models, combining SAM 3D [24] for segmentation and mesh extraction with FoundationPose [25] for 6D pose tracking.

To evaluate generalization, we introduce *DexToolBench*, a benchmark of daily tool-use behaviors, both in simulation and in the real-world. Each task is paired with a human video demonstration, and the robot is evaluated on its ability to follow the demonstrated tool trajectory. Notably, *SimToolReal* is trained exclusively on procedurally-generated primitives with randomly sampled goal pose trajectories. Yet, it transfers zero-shot to the trajectory following tasks in *DexToolBench* and matches or outperforms specialist policies trained on a single object instance and task. Our contributions:

1) We propose *SimToolReal*, a unified RL framework that frames dexterous tool use as manipulating a tool through a sequence of desired poses. Training a single policy under this objective induces dexterous skills, such as stable grasping and in-hand reorientation. This policy is used zero-shot for tool-use without task-specific engineering.
2) We develop an object-centric perception pipeline that estimates the tool's 6D pose and a coarse 3D grasp bounding box using vision foundation models, enabling zero-shot sim-to-real deployment of the RL policy.
3) We introduce *DexToolBench*, a benchmark of dexterous tool-use tasks. We show that *SimToolReal* achieves strong zero-shot performance on this benchmark over 120 real-world rollouts spanning 24 tasks, 12 object instances, and 6 tool categories. We outperform prior methods using fixed grasps and motion retargeting by 37% in task progress while matching the performance of specialists.

II. Related Work

Imitation Learning from Teleoperated Data. Imitation learning is the predominant approach for training manipulation policies [26], [27], [28], relying on demonstrations collected via motion capture gloves [29], [30], VR devices [31], [32], or wearable exoskeletons [33], [34], [35]. However, the human-to-robot correspondence gap and lack of haptic feedback make collecting high-quality dexterous data difficult [3], [4]. Shared-autonomy methods [36], [37], [38] partially address this but remain object-blind at execution. In contrast, we focus on fully autonomous policies.

Learning from Human Video. Kinematic retargeting pipelines translate human hand motions into robot actions [39], [40], but struggle with contact-rich interactions. Functional retargeting methods train RL policies to repro-

duce demonstrated object motion [23], [22], [41], but require per-demonstration RL training. Our approach imitates object motion from a single video without additional training.

Sim-to-Real Learning. Policies learned in simulation can acquire dexterous skills, but transferring them requires substantial environment shaping [13], including per-task reward design and accurate object models. This has limited sim-to-real approaches to narrow skill subsets [15], [17], [20]. Inspired by recent work unifying RL policies across embodiments via procedural generation [42], [43], we train a single policy across generated primitives with a universal goal-reaching reward, inducing dexterous skills that transfer to diverse real-world tools.

III. SimToolReal

We propose *SimToolReal*, a sim-to-real RL framework for zero-shot, dexterous tool manipulation. Our key insight is to view tool-use tasks through an *object-centric lens*: a wide range of tasks can be specified as manipulating a tool through a sequence of goal poses. This reduces the problem to learning a single goal-reaching RL policy in simulation that manipulates procedurally generated tool primitives toward random goal poses.

A. Problem Formulation

Let $\boldsymbol{o}_t \in SE(3)$ denote the current object pose at time t, $\boldsymbol{s}_t$ denote robot proprioception, and $\boldsymbol{\phi}$ denote a coarse object descriptor. Given a goal pose $\boldsymbol{g} \in SE(3)$, we learn a policy $\boldsymbol{a}_t = \pi_\theta(\boldsymbol{s}_t, \boldsymbol{o}_t, \boldsymbol{\phi}, \boldsymbol{g})$ that outputs joint position targets. Task execution reduces to repeatedly reaching the current goal pose and advancing to the next. We assume access to a sequence of goal poses $\{\boldsymbol{g}^k\}_{k=1}^K$ with $\boldsymbol{g}^k \in SE(3)$, extracted from a human demonstration video. During execution, we advance to the next goal when $d(\boldsymbol{o}_t, \boldsymbol{g}^k) < \epsilon$.

B. Training a General-Purpose Goal-Reaching Policy

Environment Setup. At the start of each episode, we place a randomly selected object on the table in a random pose and sample a sequence of goal poses $\{\boldsymbol{g}^k\}_{k=1}^K$. The robot must grasp the object and manipulate it to reach each goal. The first goal is sampled randomly in the reachable workspace; subsequent goals are sampled close to the previous one for smooth motion.

Reward Function. We optimize [44]:

$$r = r_{\text{smooth}} + r_{\text{grasp}} + \mathbb{I}_{\text{grasped}} r_{\text{goal}}. \tag{1}$$

r_{smooth} encourages smooth actions, r_{grasp} encourages grasping, and after the object is grasped, r_{goal} drives goal-pose reaching:

$$r_{\text{goal}} = \max(d^* - d(\boldsymbol{o}_t, \boldsymbol{g}), 0) + B_{\text{succ}}\, \mathbb{I}[d(\boldsymbol{o}_t, \boldsymbol{g}) < \epsilon]\,, \tag{2}$$

consisting of a dense progress term and a sparse success bonus. Following [45], [44], we represent each pose using $D = 4$ keypoints and define $d(\boldsymbol{o}_t, \boldsymbol{g}) = \max_i \|\boldsymbol{o}_{t,i} - \boldsymbol{g}_i\|$. See Appendix D for details.

Procedural Generation. We generate tool-like primitives as combinations of a *handle* and a *head* with varying cylinder and cuboid geometries (Fig. 2). We also randomize mass distributions. Details in Appendix E.

Object-Centric Inputs. The policy receives only the current 6D tool pose and a coarse 3D grasp-region bounding box (center + extents in the object frame). We use an LSTM backbone so the policy can implicitly infer latent physical properties from interaction history, following the spirit of [15], [46].

We train using SAPG [47] with domain randomization and an asymmetric actor-critic [48]. Full training details are provided in Appendix G.

C. Real-World Deployment

At test time, we process a human RGB-D video with SAM 3D [24] to extract a 3D mesh and grasp bounding box, and FoundationPose [25] for 6D pose tracking at 30 Hz. The policy is run in closed loop, conditioning on proprioception, current object pose, grasp bounding box, and the current goal pose. In contrast to [23], our approach does not require access to the real-world object during training, enabling zero-shot generalization. See Appendix H for details and Fig. 5 in the Appendix for an overview.

D. DexToolBench

We introduce *DexToolBench*: a real-world dexterous manipulation benchmark of challenging tool-use tasks paired with digital-twin simulation environments. It consists of 24 daily tool-use tasks across 12 unique object instances in 6 categories (hammer, marker, eraser, brush, spatula, screwdriver), requiring grasping from flat surfaces, in-hand rotations, and environment interactions. Each task is defined by an RGB-D human video processed through our perception pipeline. See Appendix I for details.

IV. Experiments

Setup. We evaluate *SimToolReal* on a 22-DoF Sharpa five-fingered left hand mounted on a 7-DoF KUKA iiwa 14 arm, using *DexToolBench* (Sec. III-D). We measure *Task Progress*: the percentage of demonstrated goal poses tracked successfully ($\epsilon = 2$cm).

A. Zero-Shot Real-World Tool-Use

We evaluate whether our single policy trained in simulation transfers zero-shot to real-world tool use. As shown in Fig. 3, our evaluation spans 24 task trajectories across 6 tool categories and 12 object instances, all unseen during training, totaling 120 real-world rollouts (5 trials each).

The policy demonstrates strong zero-shot generalization across tools with diverse masses and geometries. *Task Progress* is highest on *eraser* (translation-dominant) and *marker* (no rotation needed, though thin geometry reduces grasp reliability). The remaining categories require substantial in-hand rotation; performance degrades on thinner tools (e.g., *flat spatula* vs. *spoon spatula*) and heavier tools (e.g., 331g *mallet* vs. 36g *claw hammer*). The most common failure modes were pose tracking loss (43.7%) and object drops (34.5%). See Appendix J for detailed analysis.

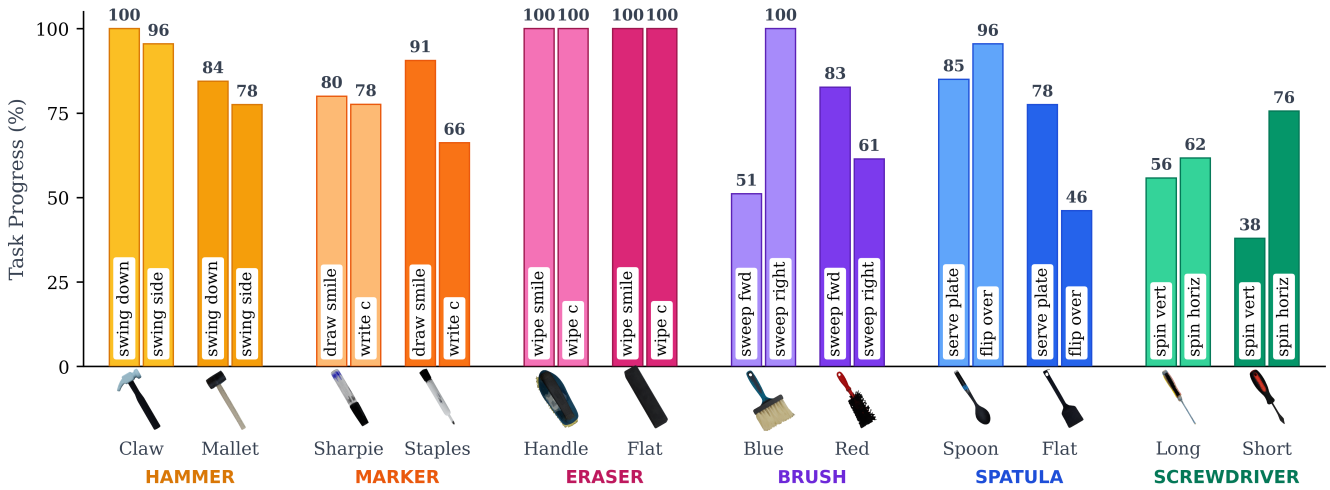

Fig. 3: **Generalization to Unseen *DexToolBench* Tools and Tasks in the Real World.** We evaluate our policy in the real world on unseen tool-use tasks in *DexToolBench*. Our evaluations span 24 unique task trajectories across 6 different object categories and 12 object instances. Each bar corresponds to 1 task trajectory on 1 object instance. We report the average *Task Progress* across 5 rollouts. Despite not being trained on these objects or trajectories, our policy demonstrates strong generalization to diverse tools of varying masses and geometries.

B. Comparisons to Baselines

Retargeting and Fixed Grasp. We compare against (i) *kinematic retargeting* [39], [23], which maps human hand motion to robot actions via IK, and (ii) *fixed grasp* [15], [49], which establishes a grasp then relies on arm motion for object rotation. On brush sweeping tasks (see Appendix Fig. 6), kinematic retargeting fails to establish stable contacts. Fixed grasp succeeds when no tool rotation is required but fails when 90° rotation is needed, as the arm collides with the table. *SimToolReal* succeeds on both variations via in-hand rotation, improving task progress by 37%. See Appendix K and L.

Specialist Policies. We compare against task-specific specialist RL policies trained on a single tool and trajectory in simulation (see Appendix Fig. 7). *SimToolReal* matches the specialist on its training setup (Obj A / Traj A), but unlike the specialist, maintains performance on novel objects (Obj B / Traj A) and trajectories (Obj A / Traj B), where the specialist degrades significantly. See Appendix M.

C. Training Objective Predicts Generalization Performance

Setup. We study how improving on our training objective: reaching random goal poses across procedurally generated primitive objects, relates to performance on our test objective: executing human-demonstrated tool-use trajectories in *DexToolBench*. Fig. 4 reports (i) training rewards on the goal-pose reaching objective on primitive objects and (ii) average *Task Progress* on *DexToolBench* trajectory following tasks in simulation, plotted against the number of environment steps.

Results. As training proceeds, the policy's reward on the goal-pose reaching objective increases steadily. In parallel, the policy's average *Task Progress* on *DexToolBench* trajectories also improves consistently. The correlation between

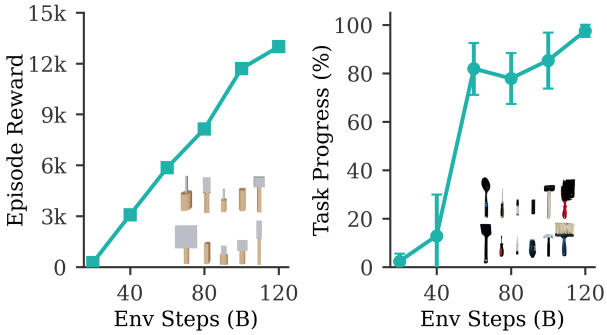

Fig. 4: **Training Objective Drives Generalization.** In simulation, we evaluate (Left) the episode reward on procedurally-generated objects and (Right) the zero-shot *Task Progress* on unseen *DexToolBench* tools on different policy checkpoints throughout training. The strong correlation between the two curves validates our core hypothesis: improving random goal-pose reaching performance on diverse object primitives drives corresponding gains in generalization to unseen tool-use tasks.

these curves indicates that optimizing our training objective is a reliable predictor of downstream generalization to tool-use behaviors, despite the mismatch between random goal-pose reaching on primitive objects during training and trajectory following on real tool objects at test time. This strong correlation suggests that our randomized primitive training effectively covers the space of skills required for real-world tool use. We ablate key RL components (SAPG, asymmetric critic) in the Appendix.

References

[1] C. Chi *et al.*, "Diffusion policy: Visuomotor policy learning via action diffusion," in *Robotics: Science and Systems*, 2023.

[2] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," *arXiv preprint arXiv:2304.13705*, 2023.

[3] 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*, vol. 10, no. 6, pp. 6416–6423, 2025.

[4] Z. Si, K. L. Zhang, Z. Temel, and O. Kroemer, "Tilde: Teleoperation for Dexterous In-Hand Manipulation Learning with a DeltaHand," in *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024.

[5] M. Arduengo, A. Arduengo, A. Colomé, J. Lobo-Prat, and C. Torras, "Human to robot whole-body motion transfer," in *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, 2021, pp. 299–305.

[6] P. Wu, Y. Shentu, Z. Yi, X. Lin, and P. Abbeel, "Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 12 156–12 163.

[7] A. M. Okamura, "Haptic feedback in robot-assisted minimally invasive surgery," *Current opinion in urology*, vol. 19, no. 1, pp. 102–107, 2009.

[8] C. Pacchierotti and D. Prattichizzo, "Cutaneous/tactile haptic feedback in robotic teleoperation: Motivation, survey, and perspectives," *IEEE Transactions on Robotics*, vol. 40, pp. 978–998, 2023.

[9] D. Morris, H. Tan, F. Barbagli, T. Chang, and K. Salisbury, "Haptic feedback enhances force skill learning," in *Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC'07)*, 2007, pp. 21–26.

[10] P. Ström, L. Hedman, L. Särnå, A. Kjellin, T. Wredmark, and L. Felländer-Tsai, "Early exposure to haptic feedback enhances performance in surgical simulator training: A prospective randomized crossover study in surgical residents," *Surgical endoscopy*, vol. 20, pp. 1383–8, 10 2006.

[11] I. Akkaya *et al.*, "Solving rubik's cube with a robot hand," *arXiv preprint arXiv:1910.07113*, 2019.

[12] T. Chen, E. Cousineau, N. Kuppuswamy, and P. Agrawal, "Vegetable peeling: A case study in constrained dexterous manipulation," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 4542–4550.

[13] Y. Park, G. B. Margolis, and P. Agrawal, "Position: Automatic environment shaping is the next frontier in rl," in *Forty-first International Conference on Machine Learning*, 2024.

[14] T. G. W. Lum *et al.*, "DextrAH-g: Pixels-to-action dexterous arm-hand grasping with geometric fabrics," in *8th Annual Conference on Robot Learning*, 2024. [Online]. Available: https://openreview.net/forum?id=S2Jwb0i7HN

[15] A. Agarwal, S. Uppal, K. Shaw, and D. Pathak, "Dexterous functional grasping," 2023. [Online]. Available: https://arxiv.org/abs/2312.02975

[16] Z. Chen *et al.*, "Clutterdexgrasp: A sim-to-real system for general dexterous grasping in cluttered scenes," *arXiv preprint arXiv:2506.14317*, 2025.

[17] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal, "Visual dexterity: In-hand reorientation of novel and complex object shapes," *Science Robotics*, vol. 8, no. 84, 2023. [Online]. Available: http://dx.doi.org/10.1126/scirobotics.adc9244

[18] OpenAI *et al.*, "Learning dexterous in-hand manipulation," 2019. [Online]. Available: https://arxiv.org/abs/1808.00177

[19] A. Handa *et al.*, "Dextreme: Transfer of agile in-hand manipulation from simulation to reality," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 5977–5984.

[20] J. Wang *et al.*, "Lessons from learning to spin "pens"," in *CoRL*, 2024.

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

[22] P. Dan *et al.*, "X-sim: Cross-embodiment learning via real-to-sim-to-real," *arXiv preprint arXiv:2505.07096*, 2025.

[23] 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," 2025. [Online]. Available: https://arxiv.org/abs/2504.12609

[24] X. Chen *et al.*, "Sam 3d: 3dfy anything in images," *arXiv preprint arXiv:2511.16624*, 2025.

[25] 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 (CVPR)*, June 2024, pp. 17 868–17 879.

[26] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, "3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations," in *Robotics: Science and Systems*, 2024.

[27] T. Lin *et al.*, "Learning visuotactile skills with two multifingered hands," *arXiv:2404.16823*, 2024.

[28] B. Cheng *et al.*, "Moe-dp: An moe-enhanced diffusion policy for robust long-horizon robotic manipulation with skill decomposition and failure recovery," *arXiv preprint arXiv:2511.05007*, 2025.

[29] K. Shaw *et al.*, "Bimanual dexterity for complex tasks," in *8th Annual Conference on Robot Learning*, 2024.

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

[31] R. Ding *et al.*, "Bunny-visionpro: Real-time bimanual dexterous teleoperation for imitation learning," 2024.

[32] A. Iyer *et al.*, "Open teach: A versatile teleoperation system for robotic manipulation," *arXiv preprint arXiv:2403.07870*, 2024.

[33] T. Tao, M. K. Srirama, J. J. Liu, K. Shaw, and D. Pathak, "Dexwild: Dexterous human interactions for in-the-wild robot policies," *arXiv preprint arXiv:2505.07813*, 2025.

[34] M. Xu *et al.*, "Dexumi: Using human hand as the universal manipulation interface for dexterous manipulation," *arXiv preprint arXiv:2505.21864*, 2025.

[35] H.-S. Fang *et al.*, "Dexop: A device for robotic transfer of dexterous human manipulation," *arXiv preprint arXiv:2509.04441*, 2025.

[36] E. Hsieh *et al.*, "Learning dexterous manipulation skills from imperfect simulations," *arXiv:2512.02011*, 2025.

[37] X. Liu, H. Wang, and L. Yi, "Dexndm: Closing the reality gap for dexterous in-hand rotation via joint-wise neural dynamics model," 2025. [Online]. Available: https://arxiv.org/abs/2510.08556

[38] Z.-H. Yin *et al.*, "Dexteritygen: Foundation controller for unprecedented dexterity," 2025. [Online]. Available: https://arxiv.org/abs/2502.04307

[39] J. Li *et al.*, "Okami: Teaching humanoid robots manipulation skills through single video imitation," in *8th Annual Conference on Robot Learning (CoRL)*, 2024.

[40] A. S. Lakshmipathy, J. K. Hodgins, and N. S. Pollard, "Kinematic motion retargeting for contact-rich anthropomorphic manipulations," *ACM Transactions on Graphics*, vol. 44, no. 2, pp. 1–20, 2025.

[41] Z. Mandi, Y. Hou, D. Fox, Y. Narang, A. Mandlekar, and S. Song, "Dexmachina: Functional retargeting for bimanual dexterous manipulation," 2025. [Online]. Available: https://arxiv.org/abs/2505.24853

[42] M. Liu, D. Pathak, and A. Agarwal, "Locoformer: Generalist locomotion via long-context adaptation," *arXiv preprint arXiv:2509.23745*, 2025.

[43] B. Ai *et al.*, "Towards embodiment scaling laws in robot locomotion," *arXiv preprint arXiv:2505.05753*, 2025.

[44] A. Petrenko, A. Allshire, G. State, A. Handa, and V. Makoviychuk, "Dexpbt: Scaling up dexterous manipulation for hand-arm systems with population based training," *CoRR*, vol. abs/2305.12127, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2305.12127

[45] A. Allshire *et al.*, "Transferring dexterous manipulation from gpu simulation to a remote real-world trifinger," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 11 802–11 809.

[46] H. Qi, A. Kumar, R. Calandra, Y. Ma, and J. Malik, "In-Hand Object Rotation via Rapid Motor Adaptation," in *Conference on Robot Learning (CoRL)*, 2022.

[47] J. Singla, A. Agarwal, and D. Pathak, "Sapg: Split and aggregate policy gradients," in *Proceedings of the 41st International Conference on Machine Learning (ICML 2024)*, ser. Proceedings of Machine Learning Research. Vienna, Austria: PMLR, July 2024.

[48] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric actor critic for image-based robot learning," in *Robotics: Science and Systems XIV, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, June 26-30, 2018*, H. Kress-Gazit, S. S. Srinivasa, T. Howard, and N. Atanasov, Eds., 2018. [Online]. Available: http://www.roboticsproceedings.org/rss14/p08.html

[49] S. Patel, S. Mohan, H. Mai, U. Jain, S. Lazebnik, and Y. Li, "Robotic manipulation by imitating generated videos without physical demonstrations," *arXiv preprint arXiv:2507.00990*, 2025.

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

[51] T. Lin *et al.*, "Learning visuotactile skills with two multifingered hands," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 5637–5643.

[52] S. P. Arunachalam, I. Güzey, S. Chintala, and L. Pinto, "Holo-dex: Teaching dexterity with immersive mixed reality," *arXiv preprint arXiv:2210.06463*, 2022.

[53] Y. Qin *et al.*, "Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system," 2024. [Online]. Available: https://arxiv.org/abs/2307.04577

[54] A. Handa *et al.*, "Dexpilot: Vision based teleoperation of dexterous robotic hand-arm system," 2019. [Online]. Available: https://arxiv.org/abs/1910.03135

[55] A. Sivakumar, K. Shaw, and D. Pathak, "Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube," *arXiv preprint arXiv:2202.10448*, 2022.

[56] P. Naughton, J. Cui, K. Patel, and S. Iba, "Respilot: Teleoperated finger gaiting via gaussian process residual learning," in *Conference on Robot Learning (CoRL)*, 2024.

[57] H. Li *et al.*, "Novaflow: Zero-shot manipulation via actionable flow from generated videos," *arXiv preprint arXiv:2510.08568*, 2025.

[58] B. Chen *et al.*, "Large video planner enables generalizable robot control," *arXiv preprint arXiv:2512.15840*, 2025.

[59] Y. Chen, C. Wang, Y. Yang, and K. Liu, "Object-centric dexterous manipulation from human motion data," in *8th Annual Conference on Robot Learning*, 2024.

[60] K. Li, P. Li, T. Liu, Y. Li, and S. Huang, "Maniptrans: Efficient dexterous bimanual manipulation transfer via residual learning," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 6991–7003.

[61] J. Mao *et al.*, "Robot learning from a physical world model," *arXiv preprint arXiv:2511.07416*, 2025.

[62] O. M. Andrychowicz *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

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

[64] S. Uppal, A. Agarwal, H. Xiong, K. Shaw, and D. Pathak, "Spin: Simultaneous perception interaction and navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 18 133–18 142.

[65] J. Ye *et al.*, "Dex1b: Learning with 1b demonstrations for dexterous manipulation," in *Robotics: Science and Systems (RSS)*, 2025.

[66] R. Singh, K. Van Wyk, P. Abbeel, J. Malik, N. Ratliff, and A. Handa, "End-to-end rl improves dexterous grasping policies," *arXiv preprint arXiv:2509.16434*, 2025.

[67] R. Singh, A. Allshire, A. Handa, N. Ratliff, and K. Van Wyk, "Dextrah-rgb: Visuomotor policies to grasp anything with dexterous hands," *arXiv preprint arXiv:2412.01791*, 2024.

[68] N. Ravi *et al.*, "Sam 2: Segment anything in images and videos," *arXiv preprint arXiv:2408.00714*, 2024. [Online]. Available: https://arxiv.org/abs/2408.00714

[69] J. Luo *et al.*, "Fmb: a functional manipulation benchmark for generalizable robotic learning," 2024. [Online]. Available: https://arxiv.org/abs/2401.08553

[70] A. Jain *et al.*, "Polaris: Scalable real-to-sim evaluations for generalist robot policies," 2025. [Online]. Available: https://arxiv.org/abs/2512.16881

[71] P. Atreya *et al.*, "Roboarena: Distributed real-world evaluation of generalist robot policies," in *Proceedings of the Conference on Robot Learning (CoRL 2025)*, 2025.

[72] W. Pumacay, I. Singh, J. Duan, R. Krishna, J. Thomason, and D. Fox, "The colosseum: A benchmark for evaluating generalization for robotic manipulation," 2024.

[73] S. James, Z. Ma, D. Rovick Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, 2020.

[74] H.-S. Fang, C. Wang, M. Gou, and C. Lu, "Graspnet-1billion: A large-scale benchmark for general object grasping," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 441–11 450.

[75] C. Li *et al.*, "Behavior-1k: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation," *arXiv preprint arXiv:2403.09227*, 2024.

[76] Y. Wang *et al.*, "Dexh2r: A benchmark for dynamic dexterous grasping in human-to-robot handover," 2025. [Online]. Available: https://arxiv.org/abs/2506.23152

[77] J. Zhang *et al.*, "Dexgraspnet 2.0: Learning generative dexterous grasping in large-scale synthetic cluttered scenes," in *8th Annual Conference on Robot Learning*.

[78] Y. Xu *et al.*, "Unidexgrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy," *arXiv preprint arXiv:2303.00938*, 2023.

[79] W. Wan *et al.*, "Unidexgrasp++: Improving dexterous grasping policy learning via geometry-aware curriculum and iterative generalist-specialist learning," *arXiv preprint arXiv:2304.00464*, 2023.

[80] I. Guzey, Y. Dai, G. Savva, R. Bhirangi, and L. Pinto, "Bridging the human to robot dexterity gap through object-oriented rewards," 2024. [Online]. Available: https://arxiv.org/abs/2410.23289

[81] R.-Z. Qiu *et al.*, "Humanoid policy ˜ human policy," *arXiv preprint arXiv:2503.13441*, 2025.

[82] G. Pavlakos, D. Shan, I. Radosavovic, A. Kanazawa, D. Fouhey, and J. Malik, "Reconstructing hands in 3D with transformers," in *CVPR*, 2024.

[83] C.-C. Hsu *et al.*, "Spot: Se(3) pose trajectory diffusion for object-centric manipulation," 2025. [Online]. Available: https://arxiv.org/abs/2411.00965

[84] Y. Zhu, A. Lim, P. Stone, and Y. Zhu, "Vision-based manipulation from single human video with open-world object graphs," *arXiv preprint arXiv:2405.20321*, 2024.

[85] S. H. Allu, J. J. P, N. Khargonkar, T. Summers, J. Yao, and Y. Xiang, "Hrt1: One-shot human-to-robot trajectory transfer for mobile manipulation," 2025. [Online]. Available: https://arxiv.org/abs/2510.21026

[86] V. Makoviychuk *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," 2021. [Online]. Available: https://arxiv.org/abs/2108.10470

[87] G. Authors, "Genesis: A generative and universal physics engine for robotics and beyond," December 2024. [Online]. Available: https://github.com/Genesis-Embodied-AI/Genesis

[88] K. Zakka *et al.*, "Mujoco playground: An open-source framework for gpu-accelerated robot learning and sim-to-real transfer." 2025. [Online]. Available: https://github.com/google-deepmind/mujoco_playground

[89] B. Yi *et al.*, "Viser: Imperative, web-based 3d visualization in python," *arXiv preprint arXiv:2507.22885*, 2025.

[90] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," University of California, San Diego, Unpublished manuscript, 2004, available at https://www.math.ucsd.edu/~sbuss/ResearchWeb/ikmethods/iksurvey.pdf.

[91] C. M. Kim*, B. Yi*, H. Choi, Y. Ma, K. Goldberg, and A. Kanazawa, "Pyroki: A modular toolkit for robot kinematic optimization," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025. [Online]. Available: https://arxiv.org/abs/2505.03728

APPENDIX

We introduced *SimToolReal*, an RL framework for zero-shot dexterous tool manipulation that generalizes to unseen tool-use tasks. Our key insight is an object-centric reduction: tool-use can be specified as manipulating a tool through a sequence of goal poses. This shifts the problem from per-task reward design to a universal objective of training goal-pose reaching policies. Training with this objective over procedurally generated primitives induces dexterous skills, such as stable grasping and in-hand reorientation, critical for tool-use.

Limitations. While our approach can track tool-use goal sequences, it does not guarantee functional task completion, especially for high-force interactions. Conditioning on object pose goals alone is environment-blind, which can lead to collisions in cluttered scenes. We also currently assume tools are rigid. Pose alone can be insufficient to describe the state of non-rigid tools (e.g., scissors). Finally, our high-level goal sequence is fixed and is not replanned dynamically.

A. Extended Related Work

Imitation Learning from Teleoperated Data. Imitation learning (IL) is the predominant approach to train robot manipulation policies [26], [27], [28]. However, IL relies on high-quality demonstrations, typically collected via human teleoperation. Many systems have been proposed to teleoperate multi-fingered dexterous hands via motion capture gloves [29], [30], VR devices [31], [32], [51], [52], or even direct camera input [53], [54], [55]. However, directly mapping hand motion to robot actions can lead to imprecise, unintuitive control [3], [4], [5]. Further, operators receive limited haptic and tactile feedback during teleoperation, which limits the collected demonstrations to simple manipulation tasks [7], [8], [9], [10].

Recent works propose wearable exoskeleton systems that physically couple the robot hand to the operator during data collection [33], [34], [35]. Still, these bulky exoskeleton systems carry a considerable burden on the operator, constrain the range of human motion, and fail to demonstrate complex skills, such as in-hand reorientations. Another line of work treats the problem of teleoperation as shared autonomy. These approaches learn structured action-spaces from human hand motion data [56] or RL in simulation [36], [37], which are used by the human operator to control the robot. Recently, DexterityGen [38] trains RL policies conditioned on target grasp poses across diverse objects. However, unlike our approach, it trains object-specific policies and distills them into a model that only conditions on hand-motion inputs. Largely, shared-autonomy methods are *object-blind* at execution time and rely on skilled operators to bridge the gap between perception and control. In contrast to all these approaches, we focus on autonomous manipulation policies without any human teleoperation effort.

Learning Dexterity from Human Video Demonstrations. Learning dexterous manipulation directly from human video demonstrations is a promising alternative to teleoperation. Kinematic motion retargeting pipelines can translate human hand motions into robot actions by reconstructing hand–object interaction [39], [40]. More recently, these pipelines have been extended to convert demonstrations synthesized by video-generation models into robot-executable trajectories [49], [57], [58]. However, because these methods primarily rely on kinematic hand-motion references, they struggle to produce actions that reliably grasp and maintain contact with tools.

To address this limitation, another line of work performs *functional retargeting*, focusing on matching the demonstrated task outcome rather than precisely matching human hand motion. For instance, hand–object 3D motion-capture can be used to train dexterous RL policies in simulation that reproduce the object motion while discovering robot-specific contact strategies [41], [59], [60]. More recently, *functional retargeting* has been extended to RGB-D human video demonstrations [23], [22] and even AI-generated videos [61]. Despite their success, these approaches typically require time-intensive RL training where the resulting policy is limited to the demonstrated manipulation task and object. In contrast, our approach can imitate object motion directly from a single RGB-D human demonstration without any additional demonstration-specific training. We can use the same policy across diverse tasks.

Sim-to-Real Learning. Policies learned in simulation can acquire dexterous manipulation behaviors without relying on human demonstrations, but transferring these skills to the real world often requires substantial *environment shaping* [13]. This includes closing the sim-to-real gap, building accurate task-specific object models, and carefully designing and tuning rewards for each behavior. Domain randomization over physical and visual parameters [19], [11], [62] and adaptation methods that infer real-world dynamics online [63], [46], [64] can reduce the sim-to-real gap significantly. Still, sim-to-real dexterous manipulation requires per-task engineering, which has led many systems to focus on narrower skill subsets such as grasping [15], [65], [14], [66], [67], in-hand object reorientation [17], [37], or dynamic object spinning [20], [21]. Recent research in locomotion has begun to unify RL policies across embodiments by leveraging large-scale simulation with procedurally generated robots and unified reward functions [42], [43]. Inspired by this direction, we train a single RL policy across procedurally generated object primitives and the universal reward function of reaching random goal poses. We show that this formulation induces highly dexterous skills (e.g., in-hand reorientation and dynamic spinning) and transfers to diverse real-world tools without task-specific engineering effort.

B. Extended Method Details

1) Object-Centric Policy Inputs: To deploy a single policy across diverse real-world tools, we need an object representation that makes learning efficient while also being feasible to extract in the real-world. Real-world tools are diverse in their geometry and physical properties, and accurately estimating detailed geometric or physical parameters from visual sensing alone is impractical. Instead, we provide the

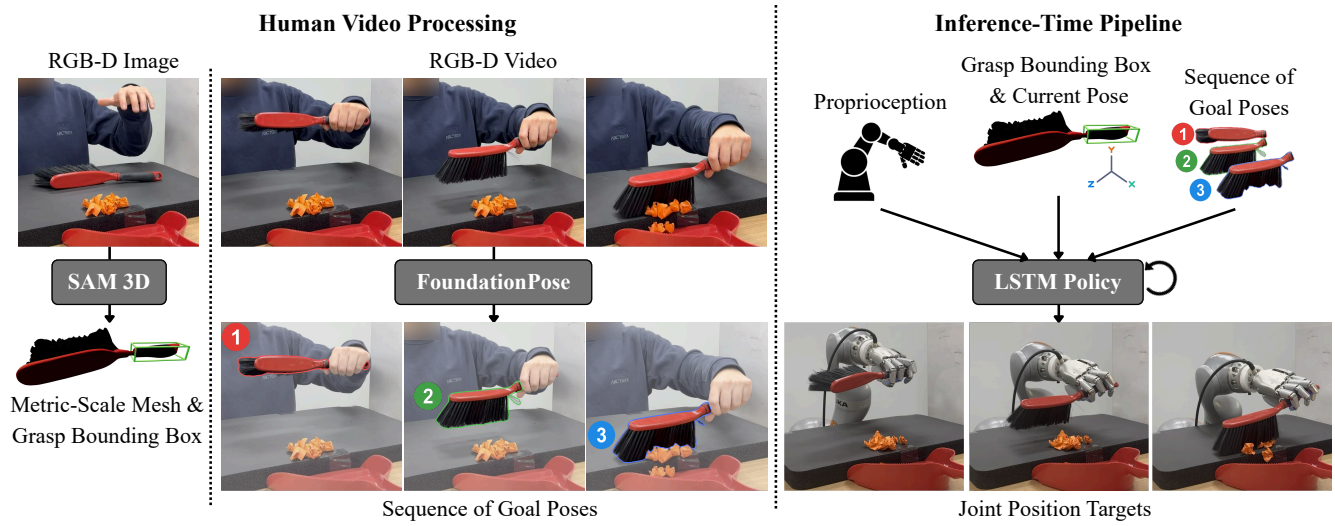

Fig. 5: **Real-World Deployment.** (Left) Human Video Processing: We collect an RGB-D human video and process it using vision foundation models. We use SAM 3D [24] to generate a metric-scale object mesh and segment a 3D grasp bounding box. Then, we use FoundationPose [25] to extract a sequence of 6D goal poses. (Right) Inference-Time Pipeline: Our LSTM policy takes in proprioception, object pose, grasp bounding box, and goal pose, and it outputs joint position targets for the 29-DoF dexterous robot (arm + hand).

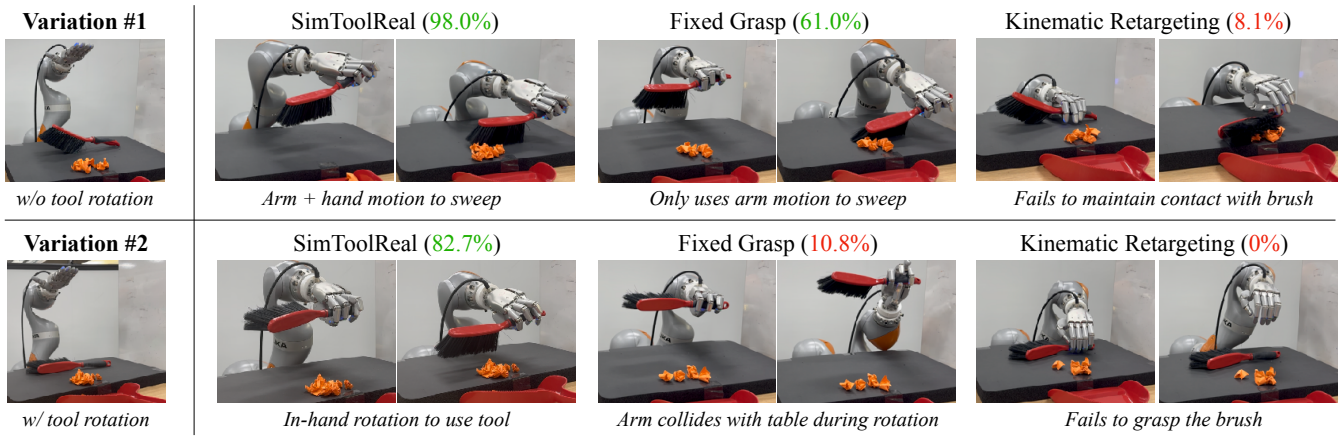

Fig. 6: **Comparison against Baselines in the Real World.** We compare *SimToolReal* against baselines on two variations of sweeping a table with a brush: with and without requiring tool rotation based on the initial states shown on the left. Average *Task Progress* is indicated in parentheses. *SimToolReal* succeeds on both variations, performing dexterous in-hand tool rotations in the harder variation. *Fixed Grasp* succeeds on the simpler variation of this task without tool rotation. However, when rotation is required, enforcing a fixed grasp causes the arm to collide with the table while tracking the target trajectory. *Kinematic Retargeting* fails to reason about contact forces, and is unable to grasp the brush in both variations.

policy only the object inputs that are reliably available at deployment: the current 6D tool pose and a coarse 3D grasp-region bounding box. The bounding box encodes the intended graspable region (center + extents in the object frame; Fig. 5) and is held fixed during an episode. We use an LSTM backbone so the policy can integrate interaction history and implicitly infer latent physical and geometric properties that are not directly observed. This design follows the spirit of prior work such as DexFunc [15] and RMA [46], which demonstrate dexterous manipulation behaviors with limited object perception to reduce the sim-to-real observation gap.

2) RL Training Details: We highlight key RL design choices essential for policy learning and sim-to-real transfer.

a) SAPG [47]: We train our policy using SAPG, a variant of PPO [50]. We find that standard PPO encounters exploration bottlenecks in massively parallel simulations. SAPG mitigates this by maintaining a population of policies to promote exploration diversity and updating the leader policy using their collective experience.

b) Domain Randomization: We apply targeted domain randomization during training to aid sim-to-real transfer. This includes observation delays, action-execution latency, noise

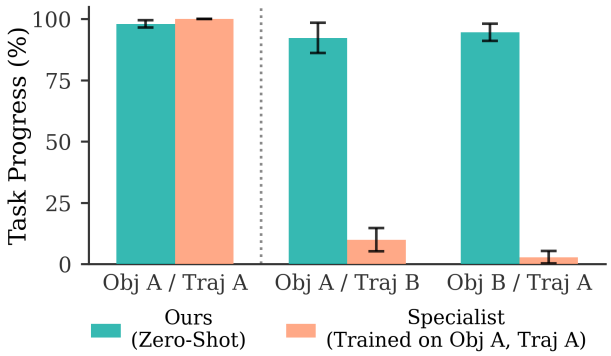

Fig. 7: **Comparison against Specialists.** We compare *SimToolReal* in simulation against specialist policies trained on a single object (Obj A) and trajectory (Traj A). We train one specialist policy for each of the 6 object categories in *DexToolBench* and report the average *Task Progress* across these categories. While the specialists succeed on their training setup (Obj A / Traj A), performance degrades under deviation in the trajectory (Obj A / Traj B) or the object (Obj B / Traj A). *SimToolReal* has high zero-shot performance across all variants, despite not being trained on these objects or trajectories.

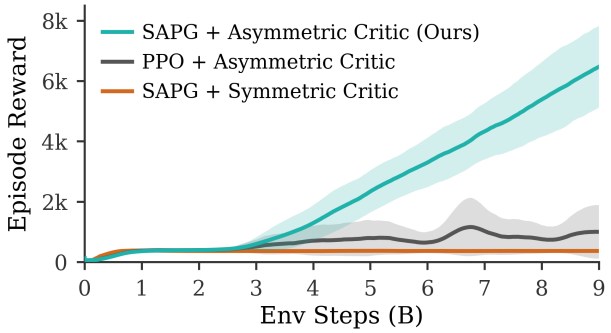

Fig. 8: **Ablation of RL Training Components.** We compare the training reward across environment steps of *SimToolReal* against ablations averaged across 5 seeds. Replacing SAPG [47] with PPO [50] or not using Asymmetric Critic [48] results in a significant performance drop, highlighting their importance.

and delay in object-pose estimates, and perturbations to the grasp-region bounding box. We additionally apply random force and torque perturbations to the object, encouraging the policy to learn strong, stable grasps. Randomization ranges and other details are provided in Appendix G.

c) Asymmetric Critic [48]: While the actor receives only the minimal object representation available at test time, we train with an asymmetric actor–critic setup in which the critic has access to privileged states in simulation. We also provide the critic with clean (noise-free, undelayed) observations to improve value estimation and stabilize training. Additional details are included in Appendix G.

3) Real-World Deployment Details: We deploy our object-centric policy in the real world to perform a wide range of tool-use tasks. Fig. 5 provides an overview. Given a third-person RGB-D view of a human demonstration, we extract (i) a 3D metric-scale object mesh and grasp region bounding box, and (ii) a sequence of target object poses that serves as the goal trajectory. During execution, we run the policy in a closed loop to reach the goal poses sequentially.

In contrast to [23], our approach does not require access to the real-world object or target trajectory during training, enabling zero-shot generalization to novel objects using demonstrations collected entirely after training.

Human Video Processing. From the first RGB-D frame, we reconstruct a metric-scale 3D mesh of the object using SAM 3D [24]. We then segment the intended grasp region using SAM 2 [68], and convert this region into a coarse 3D bounding box that is provided as part of the policy input. Finally, we run FoundationPose [25] on the RGB-D video conditioned on the extracted mesh to obtain the 6D object-pose trajectory. We process this trajectory by downsampling to 3 Hz. We use this processed trajectory as the goal sequence at test time. See Appendix H for implementation details.

Inference-time Object Tracking. During inference, we estimate the current 6D object pose with an update frequency of 30 Hz. For live pose tracking, we use FoundationPose [25] with RGB-D observations from a third-person camera view combined with the 3D mesh extracted from the human video. At each control step, the policy is conditioned on proprioception, the current object pose, the (fixed) grasp-region bounding box, and the current goal pose from the demonstration trajectory. We initialize the goal as the first pose in the sequence, switching it to the next goal whenever the pose error $d(\boldsymbol{o}_t, \boldsymbol{g})$ falls within a tolerance ϵ. The episode ends when the final goal is reached or upon failure (e.g., loss of grasp or tracking).

4) DexToolBench Details: Existing robot manipulation benchmarks primarily focus on robots with parallel-jaw grippers [69], [70], [71], [72], [73], [74], [75] or dexterous grasping [65], [76], [77], [78], [79]. *DexToolBench* is a real-world, dexterous manipulation benchmark of challenging tool-use tasks paired with digital-twin simulation environments. These tasks require grasping tools from flat surfaces, in-hand rotations to functional configurations, and environment interactions. This benchmark consists of 24 daily tool-use tasks, 12 unique object instances across 6 categories.

Each task is defined by an RGB-D human video, captured via a ZED 1 stereo camera. We process each video with our perception pipeline (Fig. 5) to extract a 3D object mesh and a 6D object pose trajectory. To facilitate reproducibility, we provide the raw RGB-D videos, processed data, simulation environments, and purchase links for the real-world tools.

Each tool instance is visualized in Fig 3. We briefly describe the skills required in each tool-use category in the benchmark:

a) Hammer: Grasp, 90° in-hand rotation, swing down or side.
b) Marker: Grasp a thin object, write on a whiteboard.
c) Eraser: Grasp, wipe marker ink on whiteboard.
d) Brush: Grasp, 90° in-hand rotation, sweep forward or right.

e) Spatula: Grasp, serve or 180° in-hand rotation and flip.
f) Screwdriver: Grasp, 90° in-hand rotation, spin in free space along vertical or horizontal axis.

See Appendix I for more details about *DexToolBench*.

C. Extended Experimental Results

1) Extended Zero-Shot Real-World Analysis: Overall, the policy demonstrates strong zero-shot generalization across tools with diverse masses and geometries. We observe the highest *Task Progress* on *eraser* trajectories, which rely primarily on translation motion rather than in-hand object rotation. Similarly, *marker* trajectories don't require in-hand rotations, but because the tool is much thinner, the grasps are less reliable. Further, the marker's small size makes it prone to pose tracking loss during occlusion.

The remaining four categories (*hammer*, *brush*, *spatula*, *screwdriver*) require substantial in-hand object rotation to reach functional tool configurations. While the policy still achieves strong *Task Progress* across these tasks, performance degrades on thinner tools (e.g., performs better on the ∼3cm thick *spoon spatula* than the ∼1cm thick *flat spatula*) and heavier tools (e.g., performs better on the 36g *claw hammer* than the 331g *mallet hammer*). We also note that tool geometry impacts task difficulty. For instance, the *red brush*'s sideways head makes *sweep forward* easier, but the *blue brush*'s frontal head makes *sweep right* easier. The *screwdriver* is the most challenging category, as the task requires both functional reorientation and continuous spinning.

Failure Analysis. The most common failure mode was pose tracking loss (43.7% of failures), followed by object drops (34.5%), failure to reach the goal pose due to incomplete in-hand rotation (18.2%), and grasp failure (3.6%). Notably, we observe strong recovery behavior when the policy makes mistakes. For example, when the object is dropped, the policy consistently attempts to re-grasp the object, provided the object remains within the workspace and pose tracking is not lost.

2) Detailed Comparison to Retargeting and Fixed Grasp: We compare *SimToolReal* with:
(i) Kinematic Retargeting: Following prior works [39], [23], [80], [81], [82], we retarget hand motion from a human video into dexterous robot actions. From an RGB-D video, we estimate 3D hand finger positions and then solve for arm and hand joint positions using an Inverse Kinematics (IK) solver. Although this method bypasses the need for RL, the resulting motion is purely kinematic and fails to account for contact interactions. See Appendix K for implementation details.
(ii) Fixed Grasp: Recent works [15], [49], [83], [84], [85] frame manipulation tasks as establishing a grasp, then subsequently fixing the grasp while moving the object. To perform any object rotations, the robot must strictly rely on arm motion. This baseline isolates the necessity of in-hand manipulation; failure here demonstrates that the arm lacks the kinematic workspace to rotate tools arbitrarily without the assistance of in-hand rotation. See Appendix L for implementation details.

To systematically evaluate the baselines, we set up two variations of *sweep forward* with the *red brush* in the real-world (see Fig. 6): (i) *Variation #1* starts with the brush top-down and does not require tool rotation, and (ii) *Variation #2* starts with the brush sideways, which necessitates a 90° tool rotation. We evaluate average *Task Progress* across 5 rollouts.

Kinematic Retargeting fails to grasp the brush in both variations as kinematic motion alone does not establish stable contacts. *Fixed Grasp* succeeds on *Variation #1* as tool rotation is not required and arm motion alone is sufficient to finish the task. Still, it has lower *Task Progress* than *SimToolReal* since it is an open-loop method and cannot react to small errors. On *Variation #2*, enforcing a fixed grasp causes the robot arm to collide with the table while tracking the target trajectory, as the optimizer is unable to find an arm motion trajectory that satisfies the tool rotation while avoiding table collision. In contrast, *SimToolReal* performs efficient in-hand object rotations to complete the task.

3) Detailed Comparison to Specialist Baseline: In simulation, we compare *SimToolReal* against task-specific specialist RL baselines trained on a single tool and task. Following [23], we train 6 specialist policies, one for each of the 6 tool-use categories in *DexToolBench*. Each specialist policy trains on a single object instance (Obj A) and task trajectory (Traj A). We evaluate both methods in simulation on (i) the training setup for the specialist: (Obj A / Traj A), (ii) the training object with a novel trajectory (Obj A / Traj B), and (iii) a novel object instance with the same trajectory (Obj B / Traj A). For each variation, we perform 10 rollouts for each of the 6 specialist policies, reporting the average *Task Progress* across all trials.

Fig. 7 summarizes these results. On the setup used for training the specialist (Obj A / Traj A), *SimToolReal* matches the specialist without training on that object or trajectory. However, the specialist's performance drops significantly under any variation from the training setup. When the object stays the same, with only a change in trajectory (Obj A / Traj B), the specialist can only track the first few goals that lift the object. The performance is lowest when the object instance is changed (e.g., from red brush to blue brush) even if the trajectory remains the same (Obj B / Traj A). This indicates that the specialist overfits to the training conditions. In contrast, *SimToolReal* shows strong zero-shot *Task Progress* across both object and trajectory variations.

4) Training Objective Predicts Generalization Performance: We study how improving on our training objective: reaching random goal poses across procedurally generated primitive objects, relates to performance on our test objective: executing human-demonstrated tool-use trajectories in *DexToolBench*. Fig. 4 reports (i) training rewards on the goal-pose reaching objective on primitive objects and (ii) average *Task Progress* on *DexToolBench* trajectory following tasks in simulation, plotted against the number of environment steps.

As training proceeds, the policy's reward on the goal-pose reaching objective increases steadily. In parallel, the policy's average *Task Progress* on *DexToolBench* trajectories also

improves consistently. The correlation between these curves indicates that optimizing our training objective is a reliable predictor of downstream generalization to tool-use behaviors, despite the mismatch between random goal-pose reaching on primitive objects during training and trajectory following on real tool objects at test time. This strong correlation suggests that our randomized primitive training effectively covers the space of skills required for real-world tool use.

5) Ablations on RL Training: We investigate which RL design decisions are critical for maximizing training performance. We focus on two key components of our RL pipeline: the choice of optimization algorithm (SAPG [47] vs. PPO [50]) and the use of privileged information (Asymmetric Critic [48] vs. standard critic). We train these variants with identical hyperparameters and environment settings, averaging results across 5 random seeds.

Fig. 8 compares the training reward curves of our full method against these ablations. *Importance of SAPG.* We observe that replacing SAPG [47] with PPO [50] leads to a significant drop in performance. While PPO suffers from exploration saturation at scale, SAPG mitigates this by training separate policies across environment chunks to increase data diversity, and then fusing gradients via importance sampling. Our results confirm that this is essential for learning complex dexterous tool-use. *Importance of Asymmetric Critic.* We observe that removing the Asymmetric Critic [48] and forcing the critic to rely solely on the same partial observations as the actor severely hinders learning. By accessing privileged simulation states, the critic learns a more accurate value function to guide policy learning. This guidance is essential for overcoming the task's partial observability.

D. Reward Function Details

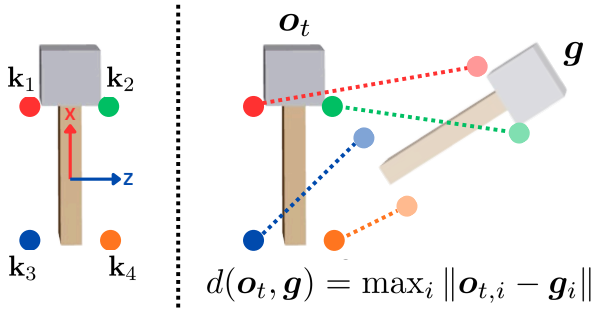

Fig. 9: **Object Keypoints.** (Left) Visualization of 4 pose keypoints in the local object frame. (Right) Visualization of the keypoint distances used to compute the distance $d(\boldsymbol{o}_t, \boldsymbol{g})$.

The reward function r has three primary components:

$$r = r_{\text{smooth}} + r_{\text{grasp}} + \mathbb{I}_{\text{grasped}} r_{\text{goal}} \tag{3}$$

where $\mathbb{I}_{\text{grasped}}$ is an indicator function that triggers once the object has been successfully grasped. We describe each reward term in further detail below:

Smoothness Reward (r_{smooth}). To promote physically plausible control and reduce hardware wear, we penalize the L_1 norm of joint velocities:

$$r_{\text{smooth}} = -\lambda_{\text{arm}} \|\dot{\mathbf{q}}_{\text{arm}}\|_1 - \lambda_{\text{hand}} \|\dot{\mathbf{q}}_{\text{hand}}\|_1 \tag{4}$$

where $\dot{\mathbf{q}}^{\text{arm}}$ and $\dot{\mathbf{q}}^{\text{hand}}$ correspond to the current velocities of the 7-DOF Kuka arm and the 22-DoF Sharpa hand, respectively.

Grasp Reward (r_{grasp}). The term r_{grasp} facilitates the transition from a neutral pose to a stable grasp:

$$r_{\text{grasp}} = r_{\text{approach}} + (1 - \mathbb{I}_{\text{grasped}}) r_{\text{lift}} \tag{5}$$

r_{approach} encourages the agent to reduce the distance between the robot's hand and the object. It rewards the agent for approaching the object rather than simply maintaining a fixed close distance. We define:

$$r_{\text{approach}} = \lambda_{\text{approach}} \max(\bar{d}^*_{\text{ft}} - \bar{d}_{\text{ft}}, 0) \tag{6}$$

where $\bar{d}_{\text{ft}}$ is the current mean distance between the fingertips and the object, and $\bar{d}^*_{\text{ft}}$ is a stateful variable that stores the minimum mean distance achieved so far in the episode. r_{lift} encourages the agent to grasp and lift the object. We define:

$$r_{\text{lift}} = \lambda_{\text{lift}} \max(z - z_{\text{init}}, 0) + \mathbb{I}[z \geq z_{\text{lifted}}] B_{\text{lifted}} \tag{7}$$

where z is the vertical position of the object, z_{init} is the initial z position of the object, z_{lifted} is the lifted threshold, and B_{lifted} is a bonus that is awarded at most once per episode when the object has been lifted. $\mathbb{I}_{\text{grasped}}$ turns true once $z \geq z_{\text{lifted}}$. This ensures that this lifting reward only applies at the start of the episode before the object has been lifted, and then is 0 for the remainder of the episode. At this point, r_{goal} takes over.

Goal-Pose Reward (r_{goal}). Once $\mathbb{I}_{\text{grasped}} = 1$, the goal-reaching term r_{goal} becomes the dominant signal to train our RL policy. As Section III-B describes, we define:

$$r_{\text{goal}} = \max(d^* - d(\boldsymbol{o}_t, \boldsymbol{g}), 0) + B_{\text{succ}} \mathbb{I}[d(\boldsymbol{o}_t, \boldsymbol{g}) < \epsilon], \tag{8}$$

Our goal-reaching reward has (i) a dense progress term based on the goal distance $d(\boldsymbol{o}_t, \boldsymbol{g})$ and (ii) a sparse success bonus B_{succ}. We maintain a stateful variable d^* storing the smallest distance achieved for the current goal (reset when a new goal is sampled), and only reward improvements in distance to avoid incentivizing static proximity. When the goal is reached, i.e., $d(\boldsymbol{o}_t, \boldsymbol{g}) < \epsilon$, the agent receives B_{succ} and we resample a new goal.

Keypoint Distance Formulation: Following [45], [44], we measure distance using D=4 object-frame keypoints $d(\boldsymbol{o}_t, \boldsymbol{g}) = \max_i \|\boldsymbol{o}_{t,i} - \boldsymbol{g}_i\|$, where $\boldsymbol{o}_{t,i}$ and $\boldsymbol{g}_i$ are the world-frame positions of the i-th keypoint under the current and goal poses, respectively. We define the following scales $\mathbf{s}^{\text{rew}} = [s_x^{\text{rew}}, s_y^{\text{rew}}, s_z^{\text{rew}}] = [0.14, 0.03, 0.03]$ (in metres), and we define the keypoints $\{\mathbf{k}_i\}_{i=1}^4$ at the following offsets:

$$\mathbf{k} \in \left\{ \begin{bmatrix} s_x^{\text{rew}}/2 \\ s_y^{\text{rew}}/2 \\ s_z^{\text{rew}}/2 \end{bmatrix}, \begin{bmatrix} s_x^{\text{rew}}/2 \\ s_y^{\text{rew}}/2 \\ -s_z^{\text{rew}}/2 \end{bmatrix}, \begin{bmatrix} -s_x^{\text{rew}}/2 \\ -s_y^{\text{rew}}/2 \\ s_z^{\text{rew}}/2 \end{bmatrix}, \begin{bmatrix} -s_x^{\text{rew}}/2 \\ -s_y^{\text{rew}}/2 \\ -s_z^{\text{rew}}/2 \end{bmatrix} \right\} \tag{9}$$

Fig. 9 visualizes our keypoint-based pose representation. Defining pose distance in this space yields a single, interpretable metric that jointly captures translation and rotation

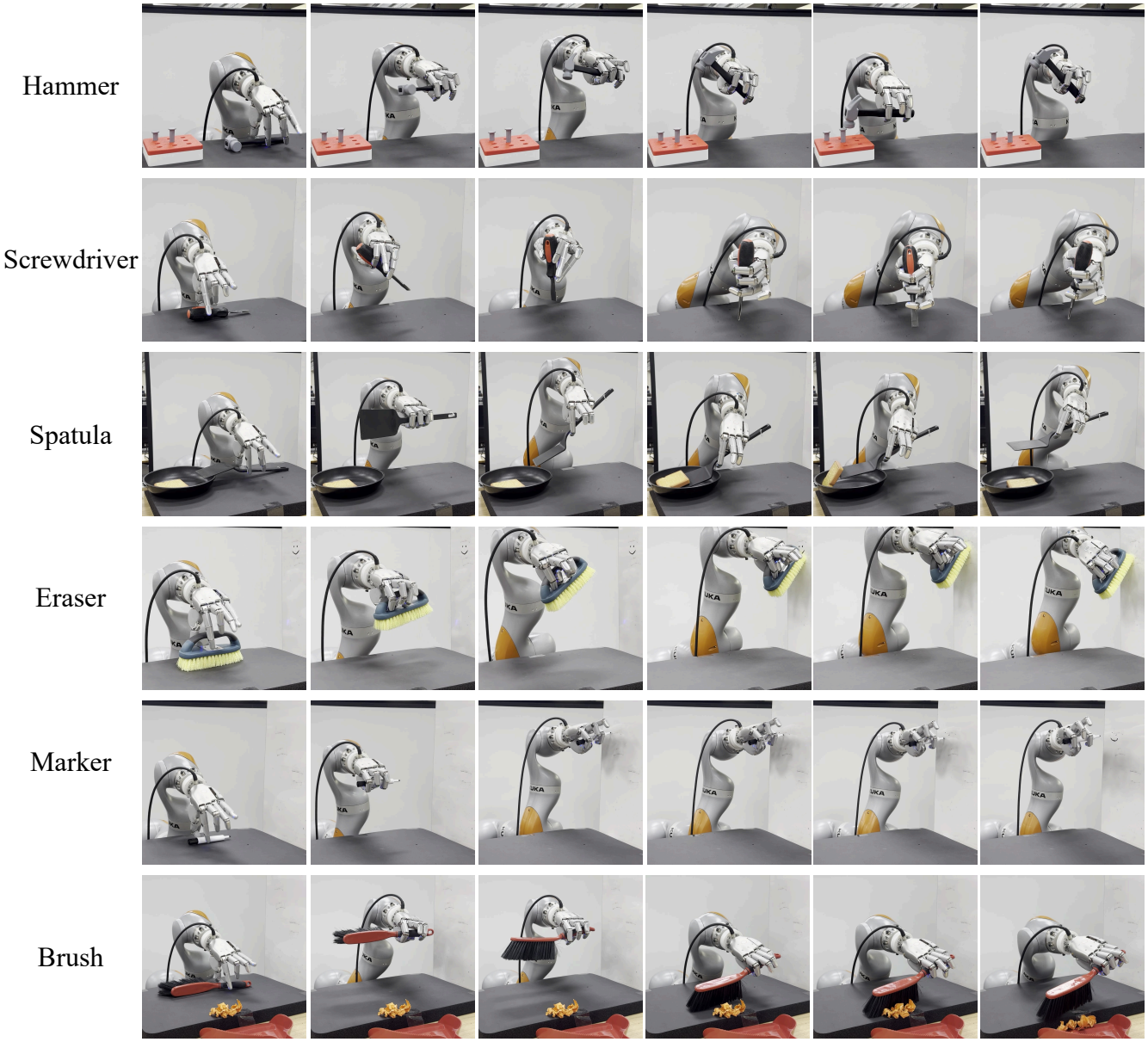

Fig. 10: **Representative Examples of *DexToolBench* Tasks.** Visual breakdown of representative tasks in *DexToolBench* across the 6 tool categories, highlighting the diversity of objects and manipulation tasks.

error. Since many tools are elongated, we set $s_x^{\text{rew}} > s_y^{\text{rew}}, s_z^{\text{rew}}$ so the reward is more sensitive to pitch and yaw errors (rotations about the y and z axes) than to roll about the long x-axis. This biases the policy toward aligning the tool's principal axis with the target pose, which is typically the dominant requirement for tool use. Using fixed relative scales across objects also ensures a consistent trade-off between translational and rotational progress for all tasks.

E. Procedural Asset Generation Details

To promote robustness to a wide range of inertial properties, we procedurally generate tool-like objects using a simple *handle–head* abstraction. This minimal design still spans a broad family of real-world handheld tools (e.g., brushes, markers, spatulas, screwdrivers, and hammers). The goal is to expose the policy to large variations in geometry and physics without relying on complex meshes that slow down simulation and training. Each tool is composed of cuboids and cylinder primitives, which ensures stable physics and fast simulation.

Tool parameterization. Each asset consists of (i) a handle and (ii) a head rigidly attached to one end of the handle. We randomize both geometry and density. For each part, we sample its length L and cross-sectional dimensions. For cuboids we use width W and height H; for cylindrical/capsule variants, W denotes the diameter and we set $H = W$.

Geometric randomizations. Each tool is a rigid union of two parts: a *handle* and a *head*. For each part, we randomly choose one of two primitive shapes: a *cuboid*

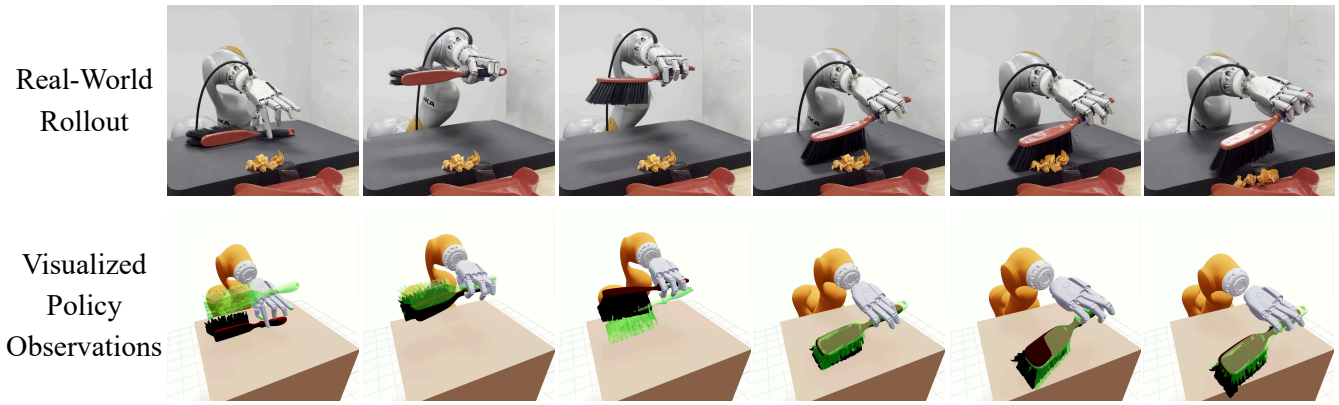

Fig. 11: **Visualization of Policy Observations during Real-World Deployment.** (Top) Image frames from a real-world rollout of the brush manipulation task. (Bottom) The corresponding visualization of the policy observations at each timestep, consisting of the robot state, estimated object pose, and goal pose (green). When the distance between the current pose and goal pose is sufficiently small, the goal pose is updated to the next pose in the goal sequence. Note that this visualization is a rendering of the policy's inputs, not a physics-based simulation.

(box) or a *capsule* (cylinder with rounded ends). We then sample the part dimensions uniformly from ranges chosen to span common handheld tools (from small brushes/markers to larger hammers). The head is attached to one end of the handle, and the head's long axis is rotated by 90° relative to the handle's long axis. For a *cuboid*, we sample a length, width, and height. For a *capsule*, we sample a length and a diameter:

Handle dimensions: length is sampled in $[5, 30]$ cm; width/height (or diameter) is sampled in $[1, 4]$ cm.

Head dimensions: length is sampled in $[1, 15]$ cm; width/height (or diameter) is sampled in $[0.5, 12]$ cm.

Physics randomizations. To systematically induce physics randomizations, we assign different densities to the handle and head. Handles are sampled with a lower density, $\rho_{\text{low}} \sim \mathcal{U}[300, 600]$ kg/m^3, approximating wood or plastics. Heads are sampled with a higher density range $\rho_{\text{high}} \sim \mathcal{U}[300, 2000]$ kg/m^3, approximating metals or dense rubber. Together with geometric variation, this density variation results in diverse center-of-mass locations (often shifted toward the head) and a broad range of rotational inertias, encouraging the policy to learn to adapt under varying physics in the real-world.

F. Simulation Training Details

1) Simulation Details: We use IsaacGym [86] for massively-parallel GPU-accelerated simulation. We build from the DexPBT Kuka Allegro Reorientation environment [44], and add substantial changes to the environment to improve dexterous tool-use, enable sim-to-real transfer, and improve generalization to unseen objects and trajectories.

Notably, many other GPU-accelerated simulators, such as Genesis [87] and MJWarp [88], could not be used because they do not currently support simulation of parallel environments with different object geometries per scene.

2) Policy Action: The RL policy outputs an action $\boldsymbol{a}_t \in \mathbb{R}^J$, where $J = 29$ represents the total number of actuated joints. We first clip the actions to the range $[-1, 1]$. We partition this action into arm components $\boldsymbol{a}_t^{\text{arm}} \in [-1, 1]^7$ and hand components $\boldsymbol{a}_t^{\text{hand}} \in [-1, 1]^{22}$. These are processed into joint position targets $\boldsymbol{q}_t^{\text{target}}$ as follows:

Arm Control (Delta). For the 7-DoF arm, we interpret the action as a relative displacement (delta) from the previous target. The intermediate target is computed as:

$$\hat{\boldsymbol{q}}_t^{\text{arm}} = \boldsymbol{q}_{t-1}^{\text{target, arm}} + k^{\text{arm}} \boldsymbol{a}_t^{\text{arm}} \tag{10}$$

where $k^{\text{arm}} = 0.025$ is a scaling factor derived from the control frequency and speed limits. We clip $\hat{\boldsymbol{q}}_t^{\text{arm}}$ to the joint limits, and then apply an exponential moving average (EMA) filter with smoothing factor α_{arm} to obtain the final $\boldsymbol{q}_t^{\text{target, arm}}$.

Hand Control (Absolute). For the 22-DoF hand, we interpret the action as an absolute target within the joint limits. We map the action range $[-1, 1]$ to the physical limits $[\boldsymbol{q}_{\text{lower}}^{\text{hand}}, \boldsymbol{q}_{\text{upper}}^{\text{hand}}]$ via an affine transformation:

$$\hat{\boldsymbol{q}}_t^{\text{hand}} = \frac{\boldsymbol{a}_t^{\text{hand}} + 1}{2} \odot (\boldsymbol{q}_{\text{upper}}^{\text{hand}} - \boldsymbol{q}_{\text{lower}}^{\text{hand}}) + \boldsymbol{q}_{\text{lower}}^{\text{hand}} \tag{11}$$

We then apply an EMA filter with smoothing factor α_{hand} to $\hat{\boldsymbol{q}}_t^{\text{hand}}$, and finally clip the result to the joint limits to ensure safety, yielding $\boldsymbol{q}_t^{\text{target, hand}}$.

In our experiments, we use $\alpha_{\text{arm}} = \alpha_{\text{hand}} = 0.1$, where $\alpha = 1.0$ means no smoothing.

3) Policy Observation: The policy observation space $\mathcal{O}$ comprises robot proprioception $\boldsymbol{s}_t$, object state $\boldsymbol{o}_t$, goal information $\boldsymbol{g}$, and the object descriptor $\boldsymbol{\phi}$. We typically provide relative representations (e.g., keypoints expressed relative to the palm) over absolute world positions to facilitate generalization across tool geometries and improve invariance to absolute world coordinates. The specific observations are:

Proprioception ($\boldsymbol{s}_t$).

- Joint positions $\boldsymbol{q}_t \in \mathbb{R}^{29}$ and velocities $\dot{\boldsymbol{q}}_t \in \mathbb{R}^{29}$.
- Previous joint position targets $\boldsymbol{q}_{t-1}^{\text{target}} \in \mathbb{R}^{29}$.
- Palm pose, consisting of world position $\boldsymbol{x}_{\text{palm}} \in \mathbb{R}^3$ and orientation quaternion $\boldsymbol{r}_{\text{palm}} \in \mathbb{R}^4$.

- Fingertip positions relative to the palm center: $\{\boldsymbol{x}_{\text{tip},j} - \boldsymbol{x}_{\text{palm}}\}_{j=1}^{5} \in \mathbb{R}^{15}$.

Object Pose, Goal Pose, and Descriptor ($\boldsymbol{o}_t, \boldsymbol{g}, \boldsymbol{\phi}$). Instead of directly inputting raw pose matrices, we represent the object and goal states using $K = 4$ keypoints. These observation keypoints are computed using the object's grasp bounding box $\boldsymbol{s} \in \mathbb{R}^3$ (length, width, height), effectively conditioning the policy on the specific tool geometry.

- Object orientation quaternion $\boldsymbol{r}_{\text{obj}} \in \mathbb{R}^4$.
- Object keypoints relative to the palm: $\{\boldsymbol{o}_{t,i} - \boldsymbol{x}_{\text{palm}}\}_{i=1}^{4} \in \mathbb{R}^{12}$.
- Keypoint errors to goal: $\{\boldsymbol{o}_{t,i} - \boldsymbol{g}_i\}_{i=1}^{4} \in \mathbb{R}^{12}$. Here, $\boldsymbol{o}_{t,i}$ and $\boldsymbol{g}_i$ correspond to the i-th keypoint of the current and goal poses, respectively.
- Object scales $\boldsymbol{s} \in \mathbb{R}^3$, representing the dimensions of the tool's graspable region (handle).

We note that while the reward function (Appendix D) uses *fixed* keypoint offsets to maintain a consistent success metric across objects, the policy observations use the *instance-specific* offsets defined by the object scales $\boldsymbol{s} = [s_x, s_y, s_z]$:

$$\mathbf{k}_{\text{obs}} \in \left\{ \begin{bmatrix} s_x/2 \\ s_y/2 \\ s_z/2 \end{bmatrix}, \begin{bmatrix} s_x/2 \\ s_y/2 \\ -s_z/2 \end{bmatrix}, \begin{bmatrix} -s_x/2 \\ -s_y/2 \\ s_z/2 \end{bmatrix}, \begin{bmatrix} -s_x/2 \\ -s_y/2 \\ -s_z/2 \end{bmatrix} \right\} \tag{12}$$

4) Asymmetric Critic State: We utilize an asymmetric actor-critic setup where the critic is trained on a privileged state space $\mathcal{S}$, while the policy acts on the restricted observation space $\mathcal{O}$. The critic state $\mathcal{S}$ contains the exact, noise-free, and instantaneous ground-truth state of the system, whereas the policy observation $\mathcal{O}$ is subjected to noise and delays to bridge the sim-to-real gap. In addition to the standard observations, the critic state includes:

- Ground-truth Velocities: Exact linear and angular velocities for the palm ($\boldsymbol{v}_{\text{palm}}, \boldsymbol{\omega}_{\text{palm}}$) and the object ($\boldsymbol{v}_{\text{obj}}, \boldsymbol{\omega}_{\text{obj}}$).
- Reward Signals: The instantaneous reward r_t and the number of successes achieved this episode, which help to improve value function estimation.
- Stateful Progress Features: Auxiliary features including the minimum fingertip-to-object distance achieved since the last reset, the minimum keypoint distance achieved since last reset, the number of environment steps that have occurred since last reset, and a binary signal $\mathbb{I}_{\text{grasped}}$ indicating if the object has been lifted.
- Noise-Free and Undelayed Object Pose: Exact object pose without noise or delays.

5) Reset and Initialization: Next, we describe the termination conditions and initialization of the environment.

Episode Termination Conditions. An environment is reset if any of the following conditions are met:

- Object Fallen Off Table: The object height drops below the table surface.
- Object Drop (Hysteresis): To penalize dropping the object after grasping, we trigger a reset if the object returns to the table surface ($z < z_{\text{init}}$) *after* the grasped flag $\mathbb{I}_{\text{grasped}}$ has been activated.
- Hand Wander: The distance between the hand and the object exceeds $1.5\,\text{m}$.
- Table Force Limits: The net force measured by the table sensor exceeds $100\,\text{N}$, preventing aggressive collisions.
- Timeout: The episode duration exceeds the maximum step limit.
- Max Consecutive Successes: The agent achieves the maximum number of consecutive successes for an episode.

Scene Initialization. At the start of each episode, the robot and object are reset to randomized states to encourage robustness:

- Robot: Joint positions are initialized to a default position with additive uniform noise.
- Object: The object is spawned above the table surface with a randomized planar position (x, y) and a random rotation quaternion $\boldsymbol{r} \in SO(3)$. We also apply small random perturbations to the table height to improve robustness to geometric calibration errors.

Goal Sampling Strategy.

1) Initial Goal ($\boldsymbol{g}_0$): The first goal is sampled uniformly from a 3D workspace volume V centered above the table. The sampling ranges relative to the table center are $x \in [-0.35, 0.35]$, $y \in [-0.1, 0.2]$, and $z \in [0.15, 0.52]$, where z is up, $-y$ is forward (wrt the robot), and x is left (wrt the robot).
2) Subsequent Goals ($\boldsymbol{g}_{k+1}$): Upon successfully reaching goal $\boldsymbol{g}_k$, we sample the next goal relative to the *previous goal*. We apply a random perturbation of up to $0.1\,\text{m}$ in position and 90° in rotation.

6) Hyperparameters: Table I shows the important hyperparameters used for the simulation environment and SAPG training.

G. Sim-to-Real Details

- Observation and action delay: We model system latency by passing observations, actions, and object states through FIFO queues and randomly sampling delayed values. To reflect the uncertainty and inference latency of vision-based object pose tracking, object pose estimates are assigned a significantly larger delay than other signals.
- Accurate robot modeling: We explicitly match the simulation's joint gains, armature, and damping to those of the physical robot.
- Sensor noise: We inject zero-mean Gaussian noise into joint velocities and object pose observations, as we found these signals to exhibit the highest measurement noise in the real world. The noise magnitudes are calibrated to match empirical levels, robustifying the policy against realistic sensor noise.
- Smooth actions: We apply exponential moving average (EMA) smoothing to joint position targets to prevent high-frequency control artifacts.
- Table height randomization: We randomize the table height at the start of each episode to improve robustness to geometric variations in the physical workspace.

TABLE I: Simulation environment and SAPG training hyperparameters.

Parameter	Value	Parameter	Value
Environment & Control		***SAPG Hyperparameters***	
Simulation / Control Frequency	120 / 60 Hz	Actor Network	LSTM[1024] + MLP[1024,1024,512,512]
Num. Environments	24576	Critic Network	MLP[1024,1024,512,512]
Episode Length	600 steps	Learning Rate	1×10^{-4}
Obj. Pos. Range (x, y)	± 10 cm	Minibatch Size	98,304
Table Height Range (z)	± 1 cm	SAPG Block Size	4096
Robot Joint Pos. Range	± 0.1 rad	Entropy Bonus Scale	0.005
Success Tolerance (ϵ)	1 cm	Discount Factor (γ)	0.99
Initial Height (z_{init})	0.63 m	GAE Parameter (τ)	0.95
Lift Threshold (z_{lifted})	0.73 m	Clip Range	0.1
Domain Randomization		***Reward Coefficients***	
Obj. Pose Noise (Trans.)	1 cm	Arm Action Penalty (λ_{arm})	0.03
Obj. Pose Noise (Rot.)	$5.0°$	Hand Action Penalty (λ_{hand})	0.003
Obj. Pose Delay Max	10 steps	Approach Scale $(\lambda_{\text{approach}})$	50.0
Action/Obs Delay Max	3 steps	Lifting Scale (λ_{lift})	20.0
Joint Vel. Obs Noise (σ)	0.1 rad/s	Lifting Bonus (B_{lifted})	300.0
Perturb. Force Scale	5.0 N	Goal Pose Scale (λ_{goal})	200.0
Perturb. Torque Scale	0.5 Nm	Success Bonus (B_{succ})	1000.0

- External force and torque perturbations: We apply random external forces and torques to the object during training to help the policy generalize to unmodeled contact dynamics and unexpected disturbances.
- Finger self-collisions: We restrict the range of the finger abduction/adduction joints to a conservative subset of their physical limits. This prevents the policy from reaching configurations that could result in potentially damaging self-collisions on the real hardware.

H. Human Video Processing Details

1) Metric-Scale Mesh and Grasp Bounding Box Acquisition: Fig. 12 illustrates our pipeline, which leverages SAM 3D [24] to extract a metric-scale mesh and grasp bounding box from a single RGB-D video demonstration.

a) **Initial Object Segmentation**: We first obtain a segmentation mask of the object from the first frame of the video. We use the first frame because it typically minimizes occlusion, as this is prior to human interaction. Although segmentation could be conditioned on a text prompt, we found that prompting SAM 2 [68] with two user-specified points on the object provided superior reliability.

b) **Metric-Scale Mesh Extraction**: Next, we generate a 3D mesh using SAM 3D [24]. Although the default SAM 3D pipeline relies on monocular depth estimation from RGB input, the resulting mesh is not metrically accurate. Since we require accurate scale for both pose tracking and grasp bounding box estimation, we modify the pipeline to directly utilize the captured depth image instead of the predicted depth.

c) **Part Segmentation (Handle vs. Head)**: To define the object's origin, we segment the mesh into a *handle* (graspable region) and a *head* (non-graspable region). This decomposition allows us to center the grasp bounding box on the handle and align the object's x-axis along the handle's primary axis (x-axis pointing from handle to head). We automate this by rendering a video of the mesh from a virtual camera circling the object. The user provides point prompts on the first frame to define the handle and head, and we use SAM 2 [68] to propagate these masks across the rendered sequence.

d) **Grasp Bounding Box Definition**: Using the rendered camera extrinsics and intrinsics, we back-project the masked depth maps to 3D to obtain separate point clouds for the handle and head. We crop the full mesh based on these point clouds. Finally, we define the grasp bounding box using the handle's geometry: the box is centered at the handle's centroid, with its x-axis oriented along the vector pointing from the handle centroid toward the head centroid.

2) Goal Pose Sequence: We use FoundationPose to extract the raw object pose trajectory from the RGB-D human video demonstration, which is collected at the camera's native frequency of 30Hz. Directly tracking this raw sequence is suboptimal for two reasons: (1) frame-to-frame pose estimation jitter can make the goal pose sequence shaky, and (2) the demonstration often includes a pre-grasp phase where the object remains stationary on the table while the human approaches it. To address these issues, we apply the following preprocessing steps.

Temporal Downsampling. We downsample the trajectory from 30Hz to 3Hz. This acts as a low-pass filter, removing high-frequency perception noise and ensuring the policy tracks the underlying smooth motion profile rather than attempting to replicate spurious artifacts in the pose estimation.

Lift-off Truncation. We automatically trim the start of the trajectory to remove the static phase where the object rests on the table. We calculate the object's vertical position z_t relative to the table surface at each timestep. The goal

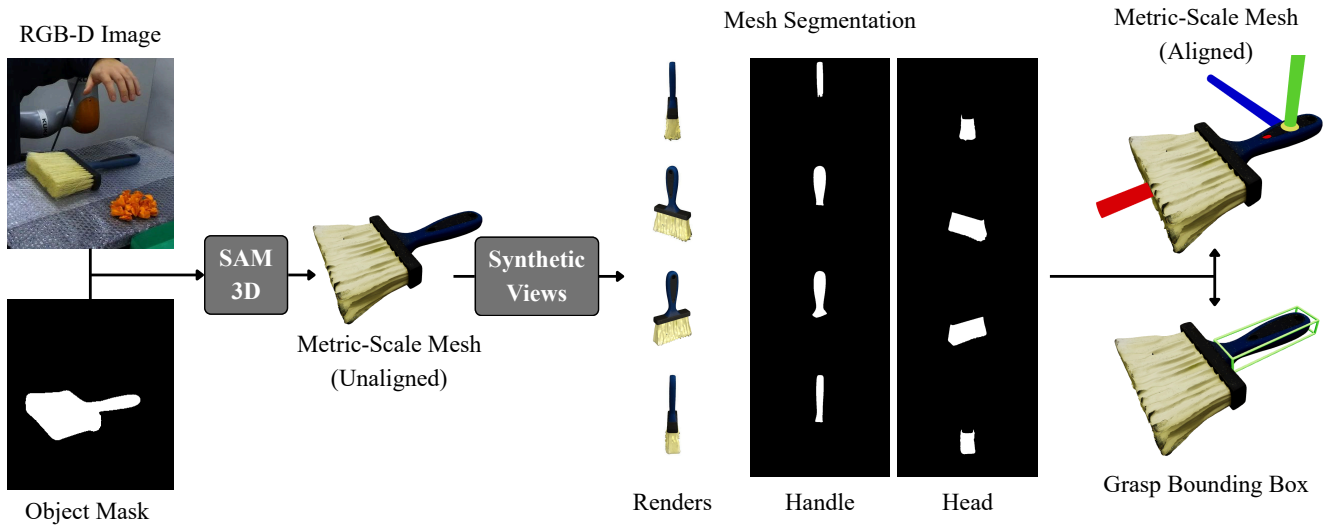

Fig. 12: **Metric-Scale Mesh and Grasp Bounding Box Acquisition.** We present a semi-automated pipeline to extract a metric-scale mesh and grasp bounding box from a single RGB-D video scan. (1) We first segment the target object in the initial frame and reconstruct a 3D mesh using SAM 3D [24], injecting the captured depth map to ensure metric accuracy. (2) To establish a canonical coordinate frame, we virtually render the mesh from multiple views and use SAM 2 [68] to segment the geometry into "handle" and "head" regions based on user prompts. (3) The final grasp bounding box is derived from the handle's geometry: it is centered on the handle, with the positive x-axis oriented toward the head, ensuring consistent alignment with policy training.

sequence is defined to start at the first frame t_{start} where the object's height exceeds a threshold $z_{\text{thresh}} = 10$cm. This ensures that the robot immediately attempts to lift and manipulate the tool, rather than trying to maintain a static pose on the table surface.

I. DexToolBench Details

Fig. 10 visualizes tasks from *DexToolBench*. Fig. 13 visualizes the real-world objects and SAM 3D [24] generated objects in *DexToolBench*.

Fig. 13: ***DexToolBench* Objects.** (Left): Real-world objects. (Right): SAM 3D [24] generated meshes of these objects.

Objects. Here, we describe the 12 object instances across the 6 object categories:

- **Hammer**:
  - *Claw Hammer*: A 3D printed hammer consisting of a thin black handle and a gray head.
  - *Mallet Hammer*: A rubber mallet consisting of a wooden handle and a heavy, cylindrical black head.
- **Marker**:
  - *Sharpie Marker*: A standard Sharpie permanent marker with a black cap.
  - *Staples Marker*: A dry-erase Staples marker with a white barrel. It is slightly thinner than the Sharpie.
- **Eraser**:
  - *Handle Eraser*: An eraser with a thin handle and yellow bristles.
  - *Flat Eraser*: An Expo eraser consisting of a rectangular foam block without a handle.
- **Brush**:
  - *Blue Brush*: A blue brush with a thick black handle and yellow bristles along the same direction as the handle.
  - *Red Brush*: A red brush with a long black handle and black bristles at a 90° angle with respect to the handle. It is lighter than the Blue brush.
- **Spatula**:
  - *Spoon Spatula*: A spoonula with a long, cylindrical black handle and a shallow oval spoon.
  - *Flat Spatula*: A black spatula with a flat rectangular blade and thin, rectangular handle.
- **Screwdriver**:
  - *Long Screwdriver*: A screwdriver with a long shaft and wooden handle. Alternating yellow, black, and red pieces of tape are added to the handle to reduce rotational symmetry.

TABLE II: **Detailed Real-World Evaluation Results.** We report the *Task Progress* (%) for each of the 5 rollouts across all 24 object-task variations. The specific tools correspond to the instances shown in Fig. 3.

Category	Instance	Trajectory	R1	R2	R3	R4	R5	Avg
Hammer	Claw	Swing Down	100	100	100	100	100	100
		Swing Side	100	100	77.5	100	100	95.5
	Mallet	Swing Down	100	100	33.3	100	88.9	84.4
		Swing Side	65.6	81.3	100	78.1	62.5	77.5
Marker	Sharpie	Draw Smile	100	100	100	100	0.0	80.0
		Write C	100	100	100	32.0	56.0	77.6
	Staples	Draw Smile	100	53.1	100	100	100	90.6
		Write C	31.0	0.0	100	100	100	66.2
Eraser	Handle	Wipe Smile	100	100	100	100	100	100
		Wipe C	100	100	100	100	100	100
	Flat	Wipe Smile	100	100	100	100	100	100
		Wipe C	100	100	100	100	100	100
Brush	Blue	Sweep Fwd	47.8	47.8	40.0	60.0	60.0	51.1
		Sweep Right	100	100	100	100	100	100
	Red	Sweep Fwd	100	100	13.5	100	100	82.7
		Sweep Right	43.2	62.2	100	42.2	59.5	61.4
Spatula	Spoon	Serve Plate	100	76.5	100	48.5	100	85.0
		Flip Over	77.5	100	100	100	100	95.5
	Flat	Serve Plate	100	75.0	78.1	56.3	78.1	77.5
		Flip Over	0.0	52.2	0.0	78.3	100	46.1
Screwdriver	Long	Spin Vert	20.9	72.1	53.5	74.4	58.1	55.8
		Spin Horiz	48.4	86.7	83.3	3.3	86.7	61.7
	Short	Spin Vert	15.4	61.5	12.8	100	0.0	37.9
		Spin Horiz	100	100	100	77.8	0.0	75.6

  - *Short Screwdriver*: A screwdriver with a short shaft and a bulbous handle with a red and black pattern.

Tasks. Here, we describe the 12 tool-use task trajectories instances across the 6 object categories:

- **Hammer**:
  - *Swing Down*: Grasp the hammer from flat on the table, rotate it by 90° into a striking configuration, swing down onto a nail 3 times.
  - *Swing Side*: Grasp the hammer from flat on the table, rotate it by 90° into a striking configuration, swing sideways onto a nail 3 times.
- **Marker**:
  - *Draw Smile*: Grasp the marker from flat on the table, move to the whiteboard, draw two dots and a smile.
  - *Write C*: Grasp the marker from flat on the table, move to the whiteboard (different location from smile), draw a C.
- **Eraser**:
  - *Wipe Smile*: Grasp the eraser from flat on the table, move to the whiteboard, erase the smile.
  - *Wipe C*: Grasp the eraser from flat on the table, move to the whiteboard (different location from smile), erase the C.
- **Brush**:
  - *Sweep Forward*: Grasp the brush from flat on the table, rotate it by 90° into a sweeping configuration, sweep forward 3 times to sweep paper balls into trash.
  - *Sweep Right*: Grasp the brush from flat on the table, rotate it by 90° into a sweeping configuration, sweep rightwards 3 times to sweep paper balls into trash.
- **Spatula**:
  - *Serve Plate*: Grasp the spatula from flat on the plate. Move to the other bowl/plate, perform a scooping motion, perform a serving motion onto the original plate.
  - *Flip Over*: Grasp the spatula upside-down with the spatula head on the pan/bowl. Rotate it by 180°, perform scooping motion, perform flipping motion.
- **Screwdriver**:
  - *Spin Vertical*: Grasp the screwdriver from flat on the table, rotate it by 90° into a vertical configuration, spin it by 360° along long axis.
  - *Spin Horizontal*: Grasp the screwdriver from flat on the table, keep it at a horizontal configuration, spin it by 360° along long axis.

Processed Data. Each task is defined by an RGB-D

human video, captured via a ZED 1 stereo camera. We process each video with our perception pipeline, described in detail in Appendix H. The processed data includes:

- Raw RGB-D video
- Object segmentation masks
- Metric-scale object meshes
- 6D object pose trajectories

Additionally, we provide:

- Visualization scripts to visualize the data in 3D space using Viser [89].
- Simulation scripts to evaluate policies on these tasks. This will support multiple robot arms and hands.
- Links to purchase the real-world objects

Task Progress. For each task, we evaluate *Task Progress*, defined as the percentage of object pose waypoints in the demonstrated trajectory that the robot successfully reached. We consider a goal pose to be reached if the distance between the object pose and goal pose $d(\boldsymbol{o}_t, \boldsymbol{g})$ is below our defined success tolerance $\epsilon = 2cm$.

This evaluation is performed *closed-loop*: the goal advances to the next waypoint only after the current waypoint is reached. This contrasts with *open-loop* playback, where goals advance at a fixed frequency regardless of the object's state. Our closed-loop protocol decouples execution speed from spatial accuracy, allowing the policy to utilize retry behaviors and execute as fast or slow as it needs to without being penalized for failing to match the exact timing profile of the demonstration.

J. Real-World Experiment Additional Analysis

Table II shows the per-rollout *Task Progress* of the results shown in Fig. 3. Fig. 11 visualizes the policy observations at each timestep during real-world deployment.

Perception Failures. We observe that object pose estimation is the most common failure mode of our system (43.7% of failures). Specifically, FoundationPose [25] struggles under three conditions common in dexterous manipulation: (1) heavy object occlusion, which is exacerbated when the robot hand manipulates small objects (e.g., the small screwdriver); (2) visual ambiguity due to rotational symmetry (e.g., the cylindrical body of the marker); and (3) low visual contrast between the object and the environment (e.g., dark bristles on a black surface). While our policy is trained to be robust to significant pose noise, it cannot recover from catastrophic tracking failures where the estimator loses the object entirely. Future work could address this by incorporating additional camera views or leveraging temporal consistency in 2D tracking to improve state estimation stability.

Manipulation Failures. Manipulation failures primarily stem from two sources: object drops (34.5%) and failure to reach the goal pose due to incomplete in-hand rotation (18.2%). Object drops were most common on heavy objects (e.g., the 331g *mallet hammer* and the 325g *blue brush*), typically occurring during reorientation or environment contact. Failure to reach the goal pose due to incomplete in-hand rotation was most common on thin objects (e.g., the ∼1cm thick *flat spatula*), as the policy would repeatedly attempt in-hand rotation but fail to manipulate the thin geometry. Grasp failures were rare (3.6%) but occasionally occurred with the marker due to its thin shape. In these instances, the marker's cylindrical shape often caused it to roll off of the table, so the policy was unable to recover.

K. Kinematic Retargeting Baseline Details

Fig. 14 visualizes the kinematic retargeting pipeline. Following [23], we perform hand pose estimation on each frame of the human video and then retarget these hand poses to a dexterous robot. Specifically, we first use HaMeR [82] to predict initial hand keypoints and mesh vertices from RGB images. Next, we refine this prediction by using the corresponding depth image and a mask of the hand (from SAM 2 [68]) to extract a 3D hand point cloud, and then aligning the HaMeR mesh to this point cloud via Iterative Closest Point (ICP) registration. Finally, we retarget the robot to these poses in two stages: we first compute the arm configuration using Damped Least Squares [90] to place the palm, and subsequently optimize the hand finger joints to reach the fingertip targets relative to the palm.

L. Fixed-Grasp Baseline Details

To run this baseline, we first use our *SimToolReal* policy to grasp and lift the object to the initial goal pose. Once the first goal pose is reached, we stop running the policy and begin planning a fixed grasp trajectory to perform the object-trajectory following task. For the hand, we maintain a fixed grasp by storing the current hand joint position targets and continuing to apply these joint position targets to the hand. For the arm, we plan a motion that moves the object along the goal trajectory, assuming its relative pose to the end-effector remains constant.

Problem Formulation. Specifically, we assume a rigid grasp where the transformation T_{EO} between the end-effector frame E and the object frame O remains constant throughout the trajectory. Let $T_{BO}^{(i)}$ denote the i-th target object pose in the robot base frame B from a goal sequence of length N. The corresponding target end-effector pose $T_{BE}^{(i)}$ is derived as:

$$T_{BE}^{(i)} = T_{BO}^{(i)} \left(T_{EO}\right)^{-1} \tag{13}$$

We test two methods for computing the arm joint trajectory $\boldsymbol{q}_{1:N}$ that reach these target end-effector poses.

- **Damped Least Squares.** For each waypoint, we compute the arm Jacobian $J(\boldsymbol{q})$ and the pose error vector $\boldsymbol{e}$ (concatenating translation and axis-angle rotation error). We compute the iterative joint update $\Delta\boldsymbol{q}$ using the Damped Least Squares method [90]:

$$\Delta\boldsymbol{q} = J^T(JJ^T + \lambda^2 I)^{-1}\boldsymbol{e} \tag{14}$$

  where the damping factor λ smooths the motion and prevents discontinuities. While this method is computationally efficient (solving in seconds), it is strictly local and does not account for environmental obstacles (e.g., the table surface or whiteboard), often leading to collisions.

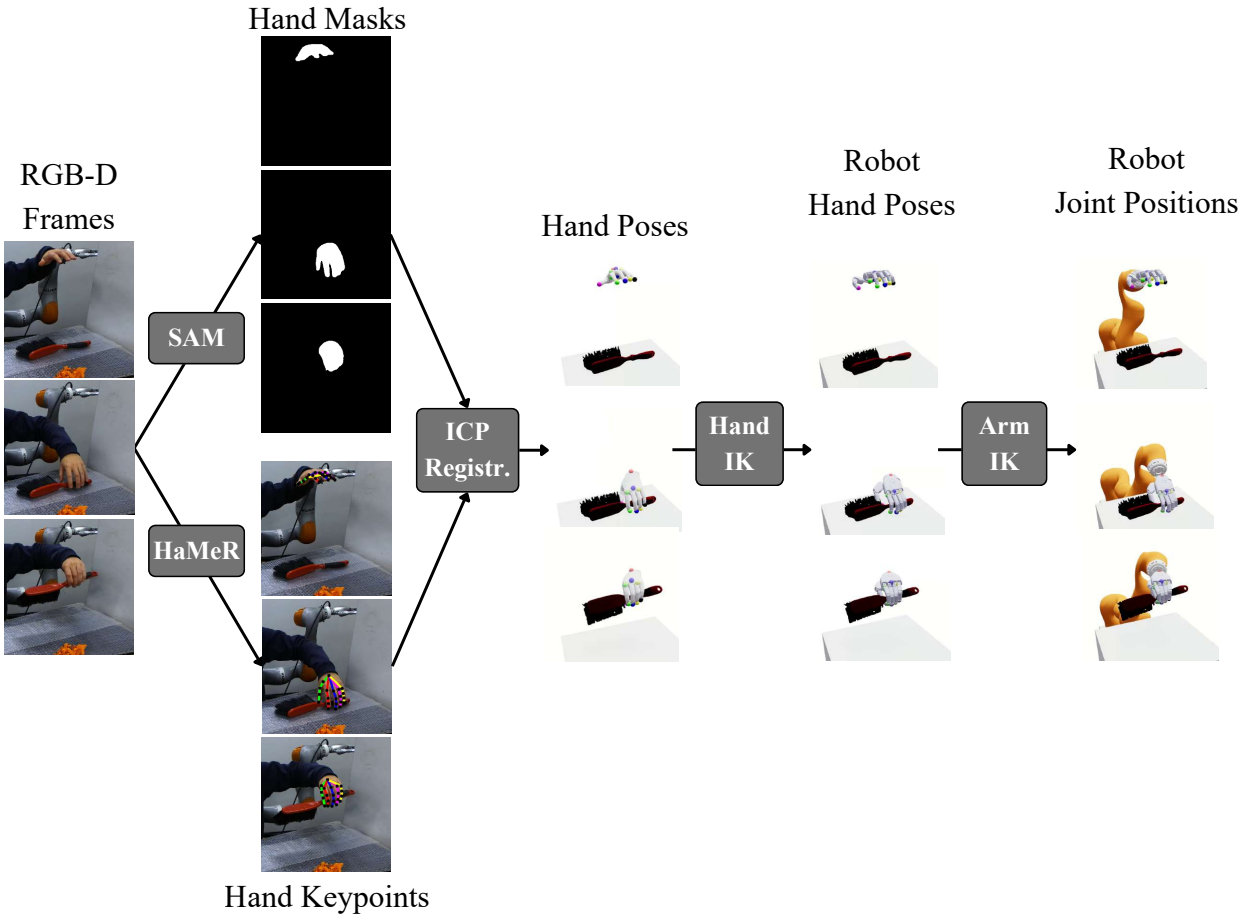

Fig. 14: **Kinematic Retargeting Pipeline.** From the RGB-D human video, we use SAM 2 [68] for hand masks and HaMeR [82] for hand pose prediction. Next, we use ICP registration to align the hand pose prediction with the segmented hand point cloud to obtain accurate 3D hand poses. Lastly, we perform IK-based retargeting of the arm and hand to match the human wrist pose and fingertip positions.

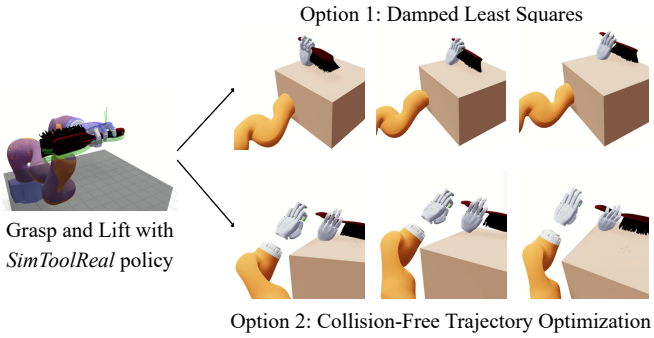

Fig. 15: **Fixed Grasp Baselines.** We first use our *SimToolReal* policy to grasp and lift the object. We then attempt to follow the trajectory using a fixed grasp. Option 1 (Damped Least Squares): Tracks the target poses but causes severe collisions with the table. Option 2 (Collision-Free Trajectory Optimization): Avoids collisions but fails to reach the target poses.

- **Collision-Free Trajectory Optimization.** To address environment collisions, we utilize PyRoki [91], which uses a JAX-based Levenberg-Marquardt solver for trajectory optimization. We extend its standard trajectory optimization formulation to optimize a trajectory with N waypoints instead of 1. This method incurs a higher computation time (~40–60 seconds per trajectory) and requires a time-consuming process of modeling the robot as a set of collision spheres.
  1) *Initialization:* We warm-start the optimization by sequentially solving collision-free Inverse Kinematics (IK) for each waypoint, seeding each step with the solution from the previous waypoint to encourage temporal consistency.
  2) *Optimization:* The solver minimizes a cost function composed of target pose error and smoothness (joint velocity) terms, subject to constraints: joint position/velocity limits, self-collision avoidance, and environment collision avoidance.

Fig. 15 visualizes the trajectories produced by both methods. We use the Collision-Free Trajectory Optimization for all reported baseline experiments, as the simple Damped

TABLE III: **Specialist Objects and Trajectories.** Objects and trajectories used for specialist policy training and evaluation.

Category	**Obj A**	**Traj A**	**Obj B**	**Traj B**
Brush	Red Brush	Sweep Forward	Blue Brush	Sweep Right
Eraser	Flat Eraser	Wipe Smile	Handle Eraser	Wipe C
Hammer	Mallet Hammer	Swing Down	Claw Hammer	Swing Side
Marker	Staples Marker	Draw Smile	Sharpie Marker	Write C
Screwdriver	Long Screwdriver	Spin Vertical	Short Screwdriver	Spin Horizontal
Spatula	Flat Spatula	Serve Plate	Spoon Spatula	Flip Over

Least Squares method frequently fails due to significant table collisions during complex object rotations.

M. Specialist Details

Fig. 16 presents a granular breakdown of the aggregate results shown in Fig. 7. While the main text reports the average performance across all categories, this figure details the specific generalization capabilities of the specialist policy trained for each of the six tool-use categories individually (e.g., Hammer, Spatula, Brush). Each specialist policy uses the same architecture, observation space, and action space as used by *SimToolReal*. It uses the same reward function, termination criteria, and episode initialization settings. The only changes are (i) *Fixed Object Geometry*: instead of training on diverse, procedurally generated primitive objects, the specialist is trained exclusively on a single object instance (Obj A), and (ii) *Fixed Task Trajectory*: instead of training on randomly sampled goal poses, the specialist is trained on a single, fixed sequence of goal poses (Traj A) extracted from a specific human demonstration.

Table III details the specific object instances and task trajectories that define Obj A, Traj A, Obj B, and Traj B for each tool category.

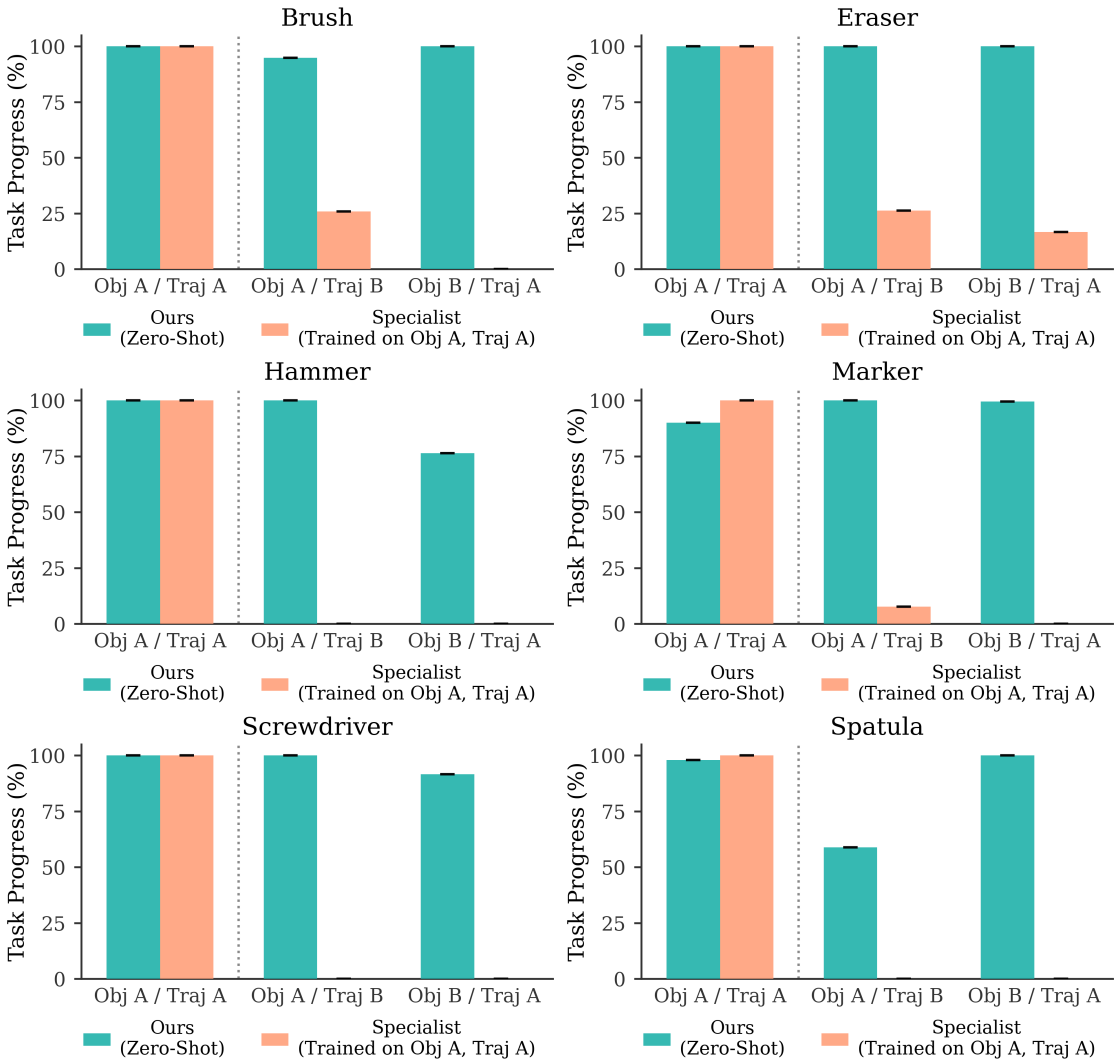

Fig. 16: **Detailed Comparison against Specialists.** We provide a breakdown of the results in Fig. 7. Each plot compares *SimToolReal* to a specialist policy trained on a single object and trajectory (Obj A / Traj A) for a single object category.