AXIS: A Growable Community-Driven Data Engine for Scalable Robot Manipulation

Anonymous Author(s)
Affiliation
Address
email

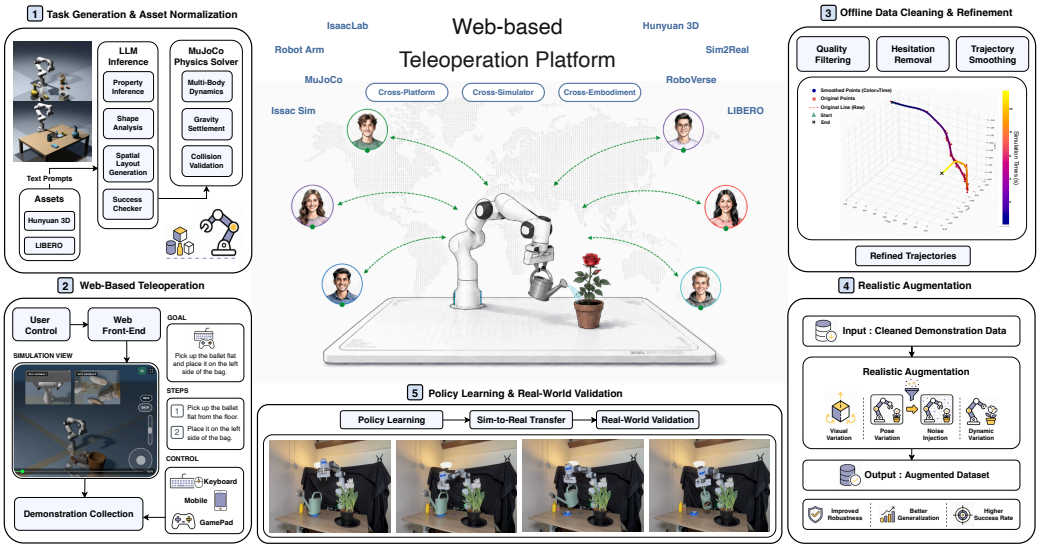

Figure 1: AXIS: A unified infrastructure bridging web-based teleoperation, GPU-accelerated realistic augmentation, and sim-to-real deployment.

Abstract: Robot manipulation policies require diverse, high-quality demonstrations, but existing data pipelines are often difficult to scale because they rely on specialized hardware, centralized operators, or fixed task suites. We present AXIS, a growable community-driven data engine and benchmark for scalable manipulation learning. AXIS collects demonstrations through browser-based MuJoCo-WASM teleoperation, generates and validates new manipulation tasks automatically, and converts community demonstrations into training-ready data through success checking, filtering, smoothing, resampling, and IsaacSim-based visual and physics augmentation. To make growth measurable, AXIS organizes data into task snapshots and evaluates policies with a fixed held-out protocol. We compare representative vision-language-action baselines under the same AXIS evaluation suite, and study scaling behavior across AXIS-100%, AXIS-50%, and AXIS-25%. These experiments test whether accessible community-collected data can support downstream policy learning, and whether increasing task coverage improves held-out manipulation generalization.

Keywords: Robot Learning, Manipulation Dataset, Data Collection

1 Introduction

General-purpose robot manipulation requires policies that can act across diverse objects, scenes, and task goals. Recent progress in imitation learning and vision-language-action modeling has shown that policy performance depends strongly on the scale, diversity, and quality of robot demonstration data[1, 2]. However, collecting such data remains difficult. Physical robot demonstrations are

Submitted to the 10th Conference on Robot Learning (CoRL 2026). Do not distribute.

expensive, teleoperation systems often require specialized hardware or local simulator installations, and many released manipulation datasets remain fixed after a one-time collection effort[3, 4]. As a result, robot learning datasets often grow much more slowly than the models they are expected to train.

A central limitation of current manipulation data pipelines is that data collection is often treated as a closed and centralized process. Expert operators or small data teams collect demonstrations on local hardware, process them offline, and release the resulting dataset as a static benchmark[4, 5]. This paradigm provides control over data quality, but it does not naturally support continuous task expansion, broad community participation, or rapid iteration as models improve. Large-scale robot learning instead requires data infrastructure that can keep growing along multiple axes: more tasks, more objects, more scenes, more contributors, more visual conditions, and more physical variations.

We argue that the next generation of robot manipulation datasets should be *growable*: not only fixed collections of trajectories, but mechanisms for continuously producing, validating, processing, and evaluating new data. Growth should also be governed, so that new tasks and demonstrations are organized into reproducible snapshots and measured under a fixed evaluation protocol. To this end, we introduce AXIS, a community-driven data engine and benchmark for scalable manipulation learning. AXIS is designed around a simple principle: make data collection broadly accessible while scaling high-quality data processing on demand. The system focuses on a Franka Research 3 arm with a parallel-jaw gripper and supports tabletop manipulation tasks including pick-and-place, stacking, pushing, pouring, articulated-object interaction, and tool-use-style manipulation. Each task includes a language instruction, parameterized scene configuration, and structured success checker.

AXIS consists of three connected layers. The infrastructure layer combines automated task generation with browser-based MuJoCo-WASM teleoperation: new tasks are generated from high-level instructions, object assets, scene layouts, and task-specific success conditions, then deployed to a shared web interface where users collect demonstrations with commodity input devices. The dataset layer converts raw community demonstrations into training-ready robot data by storing trajectories in a unified format and applying success validation, filtering, static-segment removal, smoothing, resampling, and IsaacSim-based visual and physics augmentation. The model layer uses the resulting dataset to train and evaluate conventional visuomotor imitation learning and vision-language-action policies under shared task definitions and success checkers. A key goal of AXIS is to make dataset growth measurable. AXIS organizes training data into progressively larger task snapshots (AXIS-25%, AXIS-50%, and AXIS-100%) and evaluates policies with a fixed held-out protocol, enabling controlled scaling studies where the policy architecture, training budget, rollout protocol, evaluation tasks, and success checkers remain fixed while only the training snapshot changes. Thus, we can study whether increasing task coverage improves held-out manipulation generalization.

We evaluate AXIS through two primary experimental protocols. First, we compare representative imitation learning and vision-language-action baselines under the same AXIS setup, testing whether the dataset provides a consistent benchmark for model comparison. Second, we study dataset scaling by training a fixed policy recipe across snapshots while keeping the evaluation suite and optimization budget fixed. Together, these experiments examine whether community-collected, processed, and augmented demonstrations can support downstream policy learning, and whether the growable design of AXIS provides a practical path toward scalable robot manipulation.

Contributions. Our contributions are threefold. First, we present AXIS, a web-based and community-driven infrastructure for scalable Franka manipulation data collection through MuJoCo-WASM teleoperation and automated task generation. Second, we construct a growable manipulation dataset with unified trajectory formatting, success validation, filtering, temporal smoothing, resampling, IsaacSim-based visual and physics augmentation, and versioned task snapshots. Third, we introduce an evaluation protocol for studying policy learning and within-embodiment dataset scaling, including model baseline comparison and controlled AXIS-25%/50%/100% snapshot experiments under a fixed held-out evaluation suite.

2 Related Work

Large-scale and crowdsourced robot manipulation data. Large-scale robot learning increasingly relies on broad manipulation corpora spanning multi-robot transfer, cross-environment trajectories, in-the-wild demonstrations, multi-embodiment data, and institutional aggregation [6, 7, 8, 9, 10, 11]. These datasets have supported scalable robot policies [1, 12], while recent benchmarks extend coverage to lifelong learning, household tasks, compositional reasoning, memory, grasping, and part-level supervision [13, 14, 15, 16, 17, 18, 19, 20]. Crowdsourced systems broaden collection [21, 22], but many resources remain fixed releases or depend on physical robots and specific setups. AXIS is complementary: it uses browser-based simulated teleoperation for community collection and organizes data into versioned snapshots, making dataset growth measurable rather than one-time release.

Teleoperation interfaces for robot data collection. Robot demonstrations are commonly collected by teleoperation, ranging from kinesthetic teaching [23] and leader–follower rigs [24, 25, 26] to commodity devices [14, 27, 28, 29, 30, 31] and wearable capture [32, 33]. These interfaces trade off precision, cost, embodiment fidelity, calibration burden, and accessibility. Web systems reduce onboarding, but prior work often targets remote physical robots [22, 34], leaving collection constrained by robot availability, maintenance, and safety; high-fidelity simulators meanwhile require heavier installation and GPU resources [35, 36]. AXIS separates collection from expensive simulation by using MuJoCo-WASM for browser teleoperation and reserving high-fidelity replay and augmentation for backend processing.

Simulation, task generation, and synthetic augmentation. Simulation supports scalable task variation, replay, and validation, with lightweight engines for interaction and prototyping [37, 38, 39, 40, 41] and GPU-oriented platforms for parallelism, rendering, and scene complexity [42, 43, 44, 45, 35, 36]. Standardized task suites define common observations and success criteria [46, 47, 48, 15, 2]; data-generation methods expand demonstrations by retargeting or synthesizing trajectories [49, 50]; and LLM/VLM-driven systems scale task or asset diversity [51, 52, 53, 54]. AXIS connects task generation, browser teleoperation, and IsaacSim augmentation within a single growable data pipeline.

Vision-language-action models and manipulation evaluation. Scaling data is closely tied to training and evaluating general manipulation policies. Diffusion-style policies and action-chunking transformers remain strong baselines, with ACT commonly used for chunked action prediction [25]. VLA models fine-tune large vision-language backbones on robot trajectories and have advanced instruction-following manipulation [55, 56, 57, 58, 59, 60, 61, 62, 63, 64]. Evaluation has likewise moved toward shared interfaces and robustness probes, including lifelong tasks, perturbation-aware extensions, sim-to-real-correlated evaluation, and combinatorial scene shifts [13, 65, 66, 67]. AXIS is complementary: rather than freezing a single suite, it releases versioned training snapshots with a fixed held-out protocol so policies can be compared under shared data, observation, and success-checker conventions as data grows.

3 Scalable Robot Data Collection

The dataset layer converts raw community demonstrations into training-ready robot trajectories through validation, filtering, refinement, and augmentation. Since demonstrations are collected from human operators rather than scripted controllers, the resulting data contains behavioral diversity in approach direction, grasp timing, correction behavior, and execution style.

Completed demonstrations are serialized into a unified trajectory format. Each trajectory contains task metadata, robot embodiment, environment identifier, simulator version, timestamps, observations, robot states, actions, and success information. Optional fields include object states, camera streams, segmentation markers, and failure labels when available. This format makes demonstrations replayable, filterable, and compatible with downstream policy learning.

Raw community demonstrations may contain jitter, stalls, corrupted frames, discontinuities, or incorrect success labels. The dataset is therefore filtered using consistency checks for missing

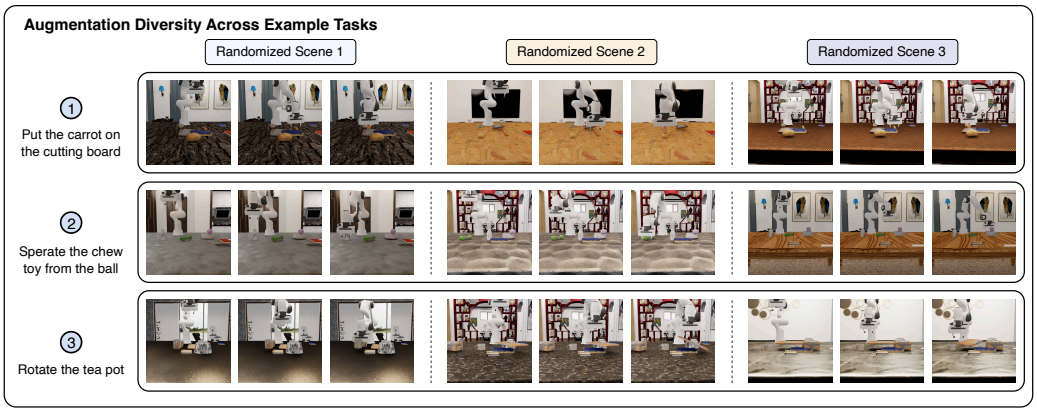

Figure 2: Visualization of augmentation diversity for three example tasks. For each task (row), we show three augmented variations (columns) obtained by randomizing appearance, lighting, environment, and initial object positions. Across augmentations, we vary materials and textures (e.g., table, floor, objects), lighting conditions (e.g., brightness and shadows), background and scene layout (e.g., walls, furniture, decorations), and initial states (e.g., object positions and orientations). This process substantially increases data diversity and improves policy robustness to environmental variations.

keys, anomalous numerical values, invalid state transitions, physically implausible frame-to-frame changes, and failed task completion. Although frontend dumps include success metadata, AXIS reuses task-specific success checkers during backend validation rather than relying solely on frontend success flags. Retained trajectories are then refined by removing static or corrupted segments, smoothing high-frequency teleoperation artifacts, and resampling to a fixed control frequency. This produces more stable action sequences while preserving the geometric structure of the demonstrated manipulation behavior.

Cleaned demonstrations are replayed in an IsaacSim backend for visual and physics augmentation. Physical randomization varies object poses, clutter configurations, mass, friction, and related dynamics parameters. Rendering randomization varies camera viewpoints, lighting, textures, and visual appearance. These perturbations are applied while preserving task semantics, expanding the training distribution without requiring additional human demonstrations for every variation. The result is a scalable pipeline that converts accessible web-collected demonstrations into higher-quality data for VLA and robot policy learning.

4 The AXIS Franka Dataset

In this section, we present the AXIS Franka Dataset and its associated validation, augmentation, and benchmarking framework.

Dataset overview. AXIS is a large-scale tabletop manipulation dataset built around a Franka Research 3 robot equipped with a parallel-jaw gripper. As summarized in Fig. 3, the current dataset snapshot contains 207 manipulation tasks, over 50k human demonstration trajectories, and more than 60k task or scene variants. Each task is specified by a language instruction, a parameterized simulation scene, task assets, and a structured success checker. Each demonstration trajectory contains synchronized multi-modal information, including RGB-D renderings from two camera views, robot states, object states, robot actions, task metadata, and success labels. This unified data schema enables the same trajectories to be used for imitation learning, replay-based verification, visual augmentation, and benchmark evaluation. The task suite covers a broad range of common tabletop manipulation behaviors, including pick-and-place, relocation, stacking, sorting, pushing, pouring, insertion, alignment, and interaction with articulated or container-like objects, and the current task distribution spans seven scene categories.

Validation and refinement. To ensure data quality, all demonstrations undergo a standardized validation pipeline that verifies trajectory completeness, task success, and physical consistency. Valid trajectories are subsequently smoothed and resampled to a unified control frequency, improving

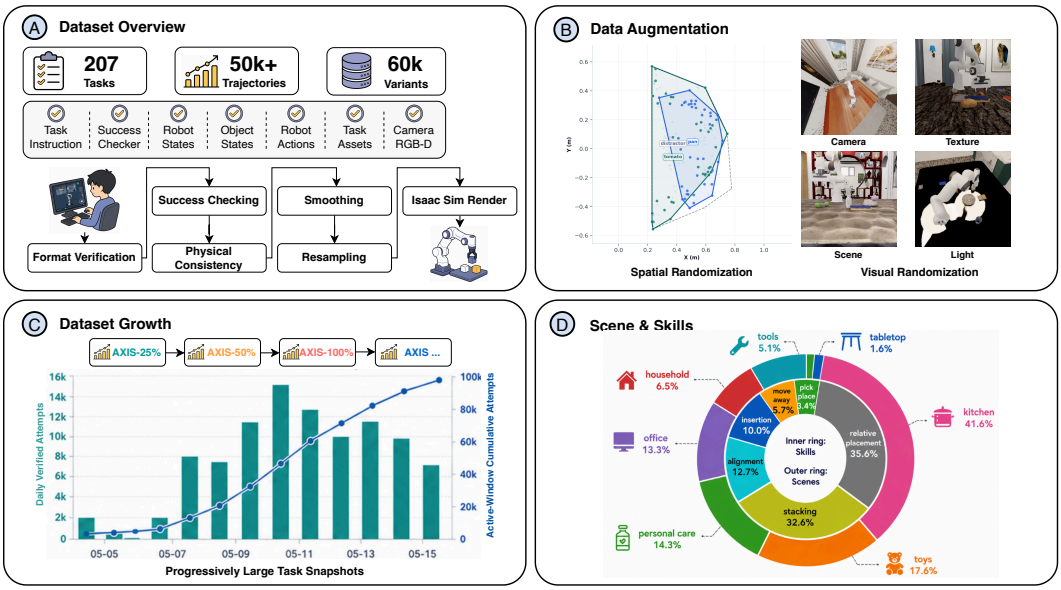

Figure 3: Overview of AXIS datasets.

Table 1: Effect of data refinement on teleoperation trajectory quality. Relative to raw teleoperation, the final smoothed-and-resampled trajectories reduce mean acceleration and mean jerk by 63.9% and 80.8%, respectively.

Data Version	Sampling Rate	Mean Acceleration	Mean Jerk	Replay Success
Raw Teleoperation	5.0 Hz	1.3539	11.5899	100.0%
Smoothed	5.0 Hz	0.6382	2.9160	91.4%
Smoothed + Resampled	20 Hz	0.4885	2.2243	86.2%

temporal consistency and reducing teleoperation noise. As shown in Table 1, the refinement process substantially reduces acceleration and jerk while maintaining a high replay success rate. Additional implementation details are provided in the Appendix **??**.

Data augmentation. To improve diversity and robustness, AXIS incorporates augmentation through spatial and visual randomization. These augmentations generate additional valid scene configurations while preserving task semantics and success conditions. As illustrated in Fig. 2, the resulting variations cover changes in object layouts, viewpoints, textures, lighting, and backgrounds, expanding the dataset to more than 60k scene variants. This diversity makes AXIS particularly useful for studying robustness and generalization under environmental variations.

Benchmarking platform. AXIS is designed as a growable benchmark for studying policy learning as data grows. The training data is organized into progressively larger task snapshots, including AXIS-25%, AXIS-50%, and AXIS-100%. Each snapshot preserves the same data format, validation pipeline, task definitions, rollout budget, and success checkers, enabling controlled scaling studies in which only the amount and diversity of training data are varied. The dataset is also continuously expandable through community-driven task and trajectory collection. In the active collection window shown in Fig. 3, the number of daily verified attempts reaches approximately 15k at its peak, and the active-window cumulative number of attempts approaches 100k. These statistics indicate that AXIS is intended to support incremental dataset growth rather than a one-time static release.

The benchmark supports two complementary evaluation modes. First, AXIS enables direct comparison across representative policy families, including conventional visuomotor imitation learning methods and vision-language-action policies. Second, AXIS supports within-embodiment scaling studies, where policies are trained on AXIS-25%, AXIS-50%, AXIS-100%, and future larger snapshots to measure whether additional task coverage improves held-out manipulation generalization.

Together, these properties position AXIS not only as a large-scale manipulation dataset, but also as a long-term benchmark for studying scalable robot learning and data-driven generalization.

5 Experiments

To evaluate AXIS as a growable data engine, we organize our experiments around three questions. (i) Does pretraining $\pi_{0.5}$ [62] on AXIS improve downstream LIBERO-Plus [65] performance, and is the improvement specific to AXIS versus a matched-volume baseline drawn from RoboCasa365 [15]? (ii) Does the improvement scale with AXIS volume, and is it largest when the downstream LIBERO fine-tuning data is scarce? (iii) On which LIBERO-Plus perturbation axes does AXIS pretraining help most, and does the pattern match the perturbations explicitly randomized by the AXIS IsaacSim augmentation pipeline?

5.1 Experimental Setup

AXIS dataset. The AXIS training corpus is collected through the browser-based MuJoCo-WASM teleoperation frontend and stored in the unified AXIS trajectory format. The robot embodiment is a Franka Research 3 arm with a parallel-jaw gripper; tasks cover tabletop manipulation behaviors including pick-and-place, stacking and sorting, articulated object interaction, pushing, pouring, and tool-use-style manipulation. Raw demonstrations are passed through the AXIS processing pipeline (success validation, static-segment removal, Savitzky–Golay smoothing, fixed-rate resampling) and then replayed in IsaacSim under randomized visual and physical conditions. We use the AXIS-100% snapshot as the largest pretraining pool, and sample uniformly without replacement at the task level to construct the 25% and 50% subsets used in scaling experiments.

Training pipeline. All conditions share the same three-stage training pipeline: start from the released $\pi_{0.5}$ checkpoint; optionally pretrain on a sim corpus (AXIS subset or RoboCasa365 subset); fine-tune on the LIBERO trajectory set provided with LIBERO-Plus. The pretraining stage uses identical optimizer settings, learning rate, batch size, and gradient-step budget across all non-vanilla conditions; the fine-tuning stage uses the $\pi_{0.5}$ default LIBERO recipe across all conditions. The RoboCasa-matched control is constructed by sampling RoboCasa365 trajectories without replacement until the trajectory count equals AXIS-100%, controlling for raw pretraining volume.

Evaluation protocol. All policies are evaluated on the LIBERO-Plus robustness suite, which extends LIBERO with systematic perturbations along seven axes (Camera, Light, Sensor Noise, Background, Layout, Language, Robot). For each (task, perturbation axis) pair we fix the rollout budget to a constant K across conditions, and report mean success rate over the rollouts. As a within-distribution sanity check, we additionally report success rate on the held-out task split of AXIS-100% itself; this confirms that AXIS pretraining does not destructively shift the policy away from the AXIS task distribution.

5.2 Main Result: AXIS Pretraining vs. Matched Sim Baseline

Our first experiment tests whether AXIS data is useful as pretraining for a downstream manipulation benchmark, and whether the benefit is specific to AXIS or generic to any matched sim corpus. We treat AXIS as *pretraining* data on top of $\pi_{0.5}$ [62], fine-tune the resulting model on the standard LIBERO trajectories, and evaluate on the LIBERO-Plus robustness benchmark [65]. This design holds the downstream task family, the fine-tuning data, and the evaluation distribution fixed for every method, isolating the effect of the pretraining corpus.

We compare four pretraining conditions. (1) $\pi_{0.5}$ **vanilla** performs the LIBERO fine-tune directly from the released $\pi_{0.5}$ checkpoint and establishes the floor. (2) $\pi_{0.5}$ **+ AXIS-25%** pretrains on a 25% random subset of AXIS-100% before LIBERO fine-tuning, providing a low-volume scaling point. (3) $\pi_{0.5}$ **+ AXIS-100%** pretrains on the full AXIS-100% snapshot and is our headline condition. (4) $\pi_{0.5}$ **+ RoboCasa-matched** pretrains on a RoboCasa365 [15] subset matched in trajectory count to AXIS-100%, controlling for the possibility that any comparably-sized Franka-sim pretraining corpus would yield similar gains. Together, conditions (1)–(3) measure whether AXIS helps and

Table 2: Main result. All conditions start from the released $\pi_{0.5}$ checkpoint, optionally pretrain on the listed corpus, then fine-tune on LIBERO trajectories using identical hyperparameters, and evaluate on LIBERO-Plus. The RoboCasa-matched control uses the same trajectory count as AXIS-100%. Per-perturbation columns report success rate on the corresponding LIBERO-Plus axis.

Pretraining	# pretrain demos	Overall ↑	Cam.	Light	Noise	B.G.	Layout	Lang.	Robot
$\pi_{0.5}$ vanilla	0	66.5	50.3	90.6	61.8	87.7	61.7	71.9	54.8
$\pi_{0.5}$ + AXIS-25%	$0.25\ N_{\mathrm{AXIS}}$	72.0	56.1	91.8	68.1	89.1	72.1	76.0	61.3
$\pi_{0.5}$ + AXIS-50%	$0.50\ N_{\mathrm{AXIS}}$	75.1	59.7	92.5	72.5	89.6	77.9	78.3	65.1
$\pi_{0.5}$ + AXIS-100% (ours)	N_{AXIS}	79.4	65.5	93.1	78.5	90.4	84.9	81.1	70.2
$\pi_{0.5}$ + RoboCasa-matched (ctrl.)	N_{AXIS}	57.5	35.2	79.5	63.2	81.7	68.0	49.2	39.4

whether more AXIS helps more; condition (4) measures whether the gain is AXIS-specific or generic to sim pretraining at this volume.

All conditions share the same downstream LIBERO fine-tuning recipe ($\pi_{0.5}$ default optimizer, batch size, schedule, and step budget), the same LIBERO trajectory set, the same LIBERO-Plus evaluation protocol, and the same number of rollouts per task and perturbation condition. Pretraining hyperparameters are also matched across conditions (2)–(4): identical learning rate, batch size, gradient-step budget, and optimizer state initialization.

The key question addressed by Table 2 is whether AXIS pretraining provides benefits beyond the existing pretraining of $\pi_{0.5}$, and whether those benefits are specific to AXIS data. AXIS pretraining consistently improves downstream performance over the vanilla $\pi_{0.5}$ baseline, with gains increasing as more AXIS data is added. The performance gap between AXIS-25% and AXIS-100% further suggests that these benefits have not yet saturated, indicating that additional AXIS data continues to improve robustness on downstream manipulation tasks. The matched-volume RoboCasa365 control tests whether these gains arise from AXIS itself or simply from additional simulation data. Despite using the same pretraining volume, RoboCasa-matched underperforms both AXIS variants across nearly all perturbation categories and on the overall benchmark. This suggests that the gains are not explained by dataset size alone, but by properties unique to the AXIS pipeline, including crowd-sourced task creation, diverse demonstrations, and large-scale environment randomization.

Overall, the results support AXIS as a scalable data engine: increasing AXIS data yields consistent downstream improvements, and those gains stem from the diversity and coverage of the AXIS pipeline rather than pretraining volume alone.

5.3 Per-Perturbation Robustness Analysis

To understand where AXIS pretraining helps and where it does not, we report the per-perturbation breakdown of the AXIS-100% condition against vanilla $\pi_{0.5}$ and the RoboCasa-matched control at the full LIBERO fine-tune budget. The LIBERO-Plus perturbation axes probe different distribution shifts: *Camera* varies viewpoint, *Light* varies illumination, *Sensor Noise* adds photometric perturbations (Gaussian noise and contrast jitter) to the camera observations, *Background* varies scene appearance and table/wall textures, *Layout* varies object initial positions and orientations, *Language* varies task wording, and *Robot* varies the robot initial pose. Different pretraining corpora are expected to help unequally across these axes depending on what their training distribution covers; AXIS's IsaacSim augmentation specifically targets camera, light, texture, and layout perturbations, so we expect the strongest relative gains on those axes.

We use Table 3 to understand where AXIS pretraining helps and where it does not. As expected from the AXIS augmentation design, the largest gains occur on visual and scene-related perturbations, including camera viewpoint, sensor noise, background appearance, and object layout, showing that IsaacSim randomization transfers effectively to corresponding robustness challenges in LIBERO-Plus. However, AXIS also improves performance on language and robot-pose variation, despite not explicitly augmenting these factors. This suggests that AXIS provides more than perturbation-specific robustness: exposure to diverse tasks, behaviors, and interaction contexts improves the underlying manipulation representations, leading to broader gains under distribution shift. Comparisons with the RoboCasa365 baseline of the same volume reinforce this conclusion. AXIS outper-

Table 3: Per-perturbation LIBERO-Plus breakdown at the full LIBERO fine-tune budget. Each row reports success rate on the corresponding perturbation axis. Δ_{van} is the absolute improvement of AXIS-100% over $\pi_{0.5}$ vanilla. $\text{AXIS}-\text{RC}$ is the difference between AXIS-100% and the matched-volume RoboCasa365 control, measuring the gain over a same-size sim corpus rather than per-trajectory superiority.

Perturbation axis	$\pi_{0.5}$ vanilla	+ AXIS-25%	+ AXIS-100%	+ RoboCasa-m.	Δ_{van}	$\text{AXIS}-\text{RC}$
Camera	50.3	56.1	65.5	35.2	+15.2	+30.3
Light	90.6	91.8	93.1	79.5	+2.5	+13.6
Sensor Noise	61.8	68.1	78.5	63.2	+16.7	+15.3
Background	87.7	89.1	90.4	81.7	+2.7	+8.7
Layout	61.7	72.1	84.9	68.0	+23.2	+16.9
Language	71.9	76.0	81.1	49.2	+9.2	+31.9
Robot	54.8	61.3	70.2	39.4	+15.4	+30.8
Overall	66.5	72.0	79.4	57.5	+12.9	+21.9

forms the control across nearly all perturbation axes, indicating that its benefits stem not only from additional simulation data but also from the diversity of environments, tasks, and demonstrations it provides. Overall, AXIS improves robustness broadly, with particularly strong gains on the visual and geometric shifts it was designed to address.

6 Discussion

Why growable datasets matter. Static datasets are valuable for benchmarking, but robot manipulation learning also benefits from data sources that can grow with model capability and observed failures. AXIS turns data collection into an iterative feedback loop: TaskGen expands semantic diversity through new manipulation problems, community teleoperation adds behavioral diversity through varied human strategies and corrections, and visual and physics augmentation expands environmental diversity through changes in appearance, scene configuration, and dynamics. In this sense, dataset growth becomes a mechanism for improving policy robustness rather than a one-time release process.

Community collection and data quality. Community-driven collection improves scale and accessibility, but introduces variation in operator skill, motion smoothness, strategy, and task completion quality. Useful variation exposes policies to diverse grasps, recoveries, and correction patterns, while harmful variation appears as corrupted states, idle segments, failed rollouts, discontinuities, or unstable actions. AXIS addresses this tension by making validation, cleaning, smoothing, replay, and augmentation core stages of dataset construction, combining broad participation with the quality control needed for downstream policy learning.

7 Conclusion

We introduced AXIS, a growable community-driven data engine and benchmark for scalable Franka manipulation learning. AXIS combines browser-based MuJoCo-WASM teleoperation, automated task generation, distributed demonstration collection, standardized trajectory processing, IsaacSim-based augmentation, and fixed-protocol evaluation. It constructs a training-ready dataset that can expand through new tasks, demonstrations, and augmented conditions, while enabling controlled comparisons across vision-language-action policies. More broadly, AXIS points toward continuously extensible robot data pipelines that connect task generation, community collection, policy training, and failure-driven improvement.

Limitations and Future Work AXIS currently focuses on simulated Franka tabletop manipulation, and sim-to-real transfer remains an open challenge. Future work will expand AXIS to more robot embodiments, richer sensing modalities, longer-horizon tasks, finer-grained annotations, and active failure-driven data collection to support more general and scalable robot learning.

References

[1] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, et al. Rt-1: Robotics transformer for real-world control at scale, 2023. URL https://arxiv.org/abs/2212.06817.

[2] H. Geng, F. Wang, S. Wei, Y. Li, B. Wang, B. An, C. T. Cheng, H. Lou, P. Li, Y.-J. Wang, Y. Liang, D. Goetting, C. Xu, H. Chen, Y. Qian, Y. Geng, J. Mao, W. Wan, M. Zhang, J. Lyu, S. Zhao, J. Zhang, J. Zhang, C. Zhao, H. Lu, Y. Ding, R. Gong, Y. Wang, Y. Kuang, R. Wu, B. Jia, C. Sferrazza, H. Dong, S. Huang, Y. Wang, J. Malik, and P. Abbeel. Roboverse: Towards a unified platform, dataset and benchmark for scalable and generalizable robot learning, 2025. URL https://arxiv.org/abs/2504.18904.

[3] J. Duan, S. Yu, H. L. Tan, H. Zhu, and C. Tan. A survey of embodied ai: From simulators to research tasks. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 6(2): 230–244, 2022.

[4] H. Choi, C. Crump, C. Duriez, A. Elmquist, G. Hager, D. Han, F. Hearl, J. Hodgins, A. Jain, F. Leve, et al. On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward. *Proceedings of the National Academy of Sciences*, 118(1):e1907856118, 2021.

[5] J. Eßer, N. Bach, C. Jestel, O. Urbann, and S. Kerner. Guided reinforcement learning: A review and evaluation for efficient and effective real-world robotics [survey]. *IEEE Robotics & Automation Magazine*, 30(2):67–85, 2022.

[6] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn. Robonet: Large-scale multi-robot learning, 2020. URL https://arxiv.org/abs/1910.11215.

[7] F. Ebert, Y. Yang, K. Schmeckpeper, B. Bucher, G. Georgakis, K. Daniilidis, C. Finn, and S. Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets, 2021. URL https://arxiv.org/abs/2109.13396.

[8] H. R. Walke, K. Black, T. Z. Zhao, Q. Vuong, C. Zheng, P. Hansen-Estruch, et al. Bridgedata v2: A dataset for robot learning at scale. In J. Tan, M. Toussaint, and K. Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 1723–1736. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/walke23a.html.

[9] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, et al. Droid: A large-scale in-the-wild robot manipulation dataset, 2025. URL https://arxiv.org/abs/2403.12945.

[10] K. Wu, C. Hou, J. Liu, Z. Che, X. Ju, Z. Yang, M. Li, Y. Zhao, Z. Xu, G. Yang, et al. Robomind: Benchmark on multi-embodiment intelligence normative data for robot manipulation. In *Robotics: Science and Systems (RSS) 2025*. Robotics: Science and Systems Foundation, 2025. URL https://www.roboticsproceedings.org/rss21/p152.pdf.

[11] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, et al. Open X-Embodiment: Robotic learning datasets and RT-X models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903, 2024. doi:10.1109/ICRA57147.2024.10611477.

[12] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In J. Tan, M. Toussaint, and K. Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

[13] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 44776–44791. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/8c3c666820ea055a77726d66fc7d447f-Paper-Datasets_and_Benchmarks.pdf.

[14] S. Nasiriany, A. Maddukuri, L. Zhang, A. Parikh, A. Lo, A. Joshi, A. Mandlekar, and Y. Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems (RSS)*, 2024.

[15] S. Nasiriany, S. Nasiriany, A. Maddukuri, and Y. Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations (ICLR)*, 2026.

[16] Y. Chen, Z. Chen, N. T. Chan, J. Chen, J. Yin, J. Shi, Y. Gao, Y.-L. Li, and J. Huo. Robohiman: A hierarchical evaluation paradigm for compositional generalization in long-horizon manipulation, 2025. URL https://arxiv.org/abs/2510.13149.

[17] Y. Dai, H. Fu, J. Lee, Y. Liu, H. Zhang, J. Yang, C. Finn, N. Fazeli, and J. Chai. Robomme: Benchmarking and understanding memory for robotic generalist policies, 2026. URL https://arxiv.org/abs/2603.04639.

[18] S. K. Srinivas, Y. Shukla, A. Arnold, and S. Chitta. Graspfactory: A large object-centric grasping dataset, 2025. URL https://arxiv.org/abs/2509.20550.

[19] P. Li, H. Geng, J. Crate, Y. Han, J. Zhang, F. Wang, C. T. Cheng, R. Dong, Y.-J. Wang, H. Lou, et al. Rose: Reconstructing objects, scenes, and trajectories from casual videos for robotic manipulation. In *NeurIPS 2025 Workshop on Bridging Language, Agent, and World Models for Reasoning and Planning*.

[20] Y. Yin, Z. Han, S. Aarya, S. Xu, J. Wang, J. Peng, A. Wang, A. Yuille, and T. Shu. Partinstruct: Part-level instruction following for fine-grained robot manipulation. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025.

[21] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, S. Savarese, and L. Fei-Fei. RoboTurk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning (CoRL)*, 2018.

[22] S. Mirchandani, M. Tang, J. Duan, J. I. Hamid, M. Cho, and D. Sadigh. Robocade: Gamifying robot data collection. *arXiv preprint arXiv:2512.21235*, 2025. doi:10.48550/arXiv.2512.21235.

[23] P. Kormushev, S. Calinon, and D. G. Caldwell. Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input. *Advanced Robotics*, 25(5):581–603, 2011. doi:10.1163/016918611X558261.

[24] P. Wu, Y. Shentu, Z. Yi, X. Lin, and P. Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators. *arXiv preprint arXiv:2309.13037*, 2023. doi:10.48550/arXiv.2309.13037.

[25] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems*, 2023. doi:10.15607/RSS.2023.XIX.016.

[26] Y. Zou, Z. Zhou, C. Shi, Z. Ye, J. Huang, Y. Ding, and B. Zhao. U-arm: Ultra low-cost general teleoperation interface for robot manipulation. *arXiv preprint arXiv:2509.02437*, 2025. doi:10.48550/arXiv.2509.02437.

[27] D. Honerkamp, H. Mahesheka, J. O. von Hartz, T. Welschehold, and A. Valada. Zero-cost whole-body teleoperation for mobile manipulation. *IEEE Robotics and Automation Letters*, 2025.

[28] C. Zhou, C. Peers, Y. Wan, R. Richardson, and D. Kanoulas. Teleman: Teleoperation for legged robot loco-manipulation using wearable imu-based motion capture. *arXiv preprint arXiv:2209.10314*, 2022. doi:10.48550/arXiv.2209.10314.

[29] A. George, A. Bartsch, and A. B. Farimani. Openvr: Teleoperation for manipulation. *SoftwareX*, 29:102054, 2025. doi:10.1016/j.softx.2025.102054.

[30] K. Lu, Y. He, C. Lu, and P. Li. I know kung fu: Synthetic dexterous hand demonstration collection via vr teleoperation. In *NeurIPS 2025 Workshop on Space in Vision, Language, and Embodied AI*.

[31] J. Wang, C.-C. Chang, J. Duan, D. Fox, and R. Krishna. Eve: Enabling anyone to train robot using augmented reality. *arXiv preprint arXiv:2404.06089*, 2024. doi:10.48550/arXiv.2404.06089.

[32] 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. doi:10.48550/arXiv.2403.07788.

[33] M. Xu, H. Zhang, Y. Hou, Z. Xu, L. Fan, M. Veloso, and S. Song. Dexumi: Using human hand as the universal manipulation interface for dexterous manipulation. *arXiv preprint arXiv:2505.21864*, 2025. doi:10.48550/arXiv.2505.21864.

[34] C.-L. Fok, F. Sun, M. Mangum, A. K. Mok, B. He, and L. Sentis. Web-based teleoperation of a humanoid robot. *arXiv preprint arXiv:1607.05402*, 2016. doi:10.48550/arXiv.1607.05402.

[35] NVIDIA. Isaac sim. `https://developer.nvidia.com/isaac/sim`, 2025. Robotics simulator, accessed 2026-03-06.

[36] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, L. Yi, A. X. Chang, L. J. Guibas, and H. Su. Sapien: A simulated part-based interactive environment. *arXiv preprint arXiv:2003.08515*, 2020. doi:10.48550/arXiv.2003.08515.

[37] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[38] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2149–2154, Sendai, Japan, Sept. 2004. doi:10.1109/IROS.2004.1389727.

[39] E. Rohmer, S. P. N. Singh, and M. Freese. Coppeliasim (formerly v-rep): A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013. doi:10.1109/IROS.2013.6696520. Available: www.coppeliarobotics.com.

[40] Robot Locomotion Group. Drake: Model-based design and verification for robotics. `https://drake.mit.edu/`, 2026. Accessed: 2026-03-06.

[41] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016. Accessed: 2026-03-06.

[42] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State. Isaac gym: High performance gpu-based physics simulation for robot learning. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021. doi:10.48550/arXiv.2108.10470. NeurIPS 2021 Datasets and Benchmarks Track.

[43] K. Zakka, B. Tabanpour, Q. Liao, M. Haiderbhai, S. Holt, J. Y. Luo, A. Allshire, E. Frey, K. Sreenath, L. A. Kahrs, C. Sferrazza, Y. Tassa, and P. Abbeel. Mujoco playground: An open-source framework for gpu-accelerated robot learning and sim-to-real transfer. *arXiv preprint arXiv:2502.08844*, 2025. doi:10.48550/arXiv.2502.08844.

[44] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem. Brax: A differentiable physics engine for large scale rigid body simulation. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021. doi:10.48550/arXiv.2106.13281. NeurIPS 2021 Datasets and Benchmarks Track.

[45] Genesis Authors. Genesis: A universal and generative physics engine for robotics and beyond. https://github.com/Genesis-Embodied-AI/Genesis, 2024. Software project, accessed 2026-03-06.

[46] J. Gu, F. Xiang, X. Li, Z. Ling, X. Liu, T. Mu, Y. Tang, S. Tao, X. Wei, Y. Yao, X. Yuan, P. Xie, Z. Huang, R. Chen, and H. Su. ManiSkill2: A unified benchmark for generalizable manipulation skills. In *International Conference on Learning Representations (ICLR)*, 2023.

[47] S. Tao, F. Xiang, A. Shukla, Y. Qin, X. Hinrichsen, X. Yuan, C. Bao, X. Lin, Y. Liu, T. kai Chan, Y. Gao, X. Li, T. Mu, N. Xiao, A. Gurha, Z. Huang, R. Calandra, R. Chen, S. Luo, and H. Su. ManiSkill3: Gpu parallelized robotics simulation and rendering for generalizable embodied AI. *arXiv preprint arXiv:2410.00425*, 2024.

[48] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, and Y. Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

[49] A. Mandlekar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox. MimicGen: A data generation system for scalable robot learning using human demonstrations. In *Conference on Robot Learning (CoRL)*, 2023.

[50] Z. Jiang, Y. Xie, K. Lin, Z. Xu, W. Wan, A. Mandlekar, L. Fan, and Y. Zhu. DexMimicGen: Automated data generation for bimanual dexterous manipulation via imitation learning. *arXiv preprint arXiv:2410.24185*, 2024.

[51] L. Wang, Y. Ling, Z. Yuan, M. Shridhar, C. Bao, Y. Qin, B. Wang, H. Xu, and X. Wang. GenSim: Generating robotic simulation tasks via large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

[52] P. Hua, M. Liu, A. Macaluso, Y. Lin, W. Zhang, H. Xu, and L. Wang. GenSim2: Scaling robot data generation with multi-modal and reasoning LLMs. In *Conference on Robot Learning (CoRL)*, 2024.

[53] Y. Wang, Z. Xian, F. Chen, T.-H. Wang, Y. Wang, K. Fragkiadaki, Z. Erickson, D. Held, and C. Gan. RoboGen: Towards unleashing infinite data for automated robot learning via generative simulation. In *International Conference on Machine Learning (ICML)*, 2024.

[54] Y. He, X. Wang, P. Li, Y. Huang, J. Masterjohn, J. Wu, L. Guibas, Y. Yang, Y. Jiang, and C. Jiang. Fishbone: From one 3d asset to a million controllable edits, 2026. URL https://arxiv.org/abs/2605.24805.

[55] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, J. Luo, Y. L. Tan, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems (RSS)*, 2024.

[56] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning (CoRL)*, 2024.

[57] M. J. Kim, C. Finn, and P. Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

[58] S. Liu, L. Wu, B. Li, H. Tan, H. Chen, Z. Wang, K. Xu, H. Su, and J. Zhu. RDT-1B: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024.

[59] Q. Li, Y. Liang, Z. Wang, L. Luo, X. Chen, M. Liao, F. Wei, Y. Deng, S. Xu, Y. Zhang, X. Wang, B. Liu, J. Fu, J. Bao, D. Chen, Y. Shi, J. Yang, and B. Guo. CogACT: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.

[60] H. Shi, B. Xie, Y. Liu, L. Sun, F. Liu, T. Wang, E. Zhou, H. Fan, X. Zhang, and G. Huang. MemoryVLA: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. *arXiv preprint*, 2025. arXiv.

[61] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[62] Physical Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[63] NVIDIA, J. Bjorck, F. C. neda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, J. Jang, Z. Jiang, J. Kautz, K. Kundalia, L. Lao, Z. Li, Z. Lin, K. Lin, G. Liu, E. Llontop, L. Magne, A. Mandlekar, A. Narayan, S. Nasiriany, S. Reed, Y. L. Tan, G. Wang, Z. Wang, J. Wang, Q. Wang, J. Xiang, Y. Xie, Y. Xu, Z. Xu, S. Ye, Z. Yu, A. Zhang, H. Zhang, Y. Zhang, Y. Zhang, L. Fan, and Y. Zhu. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[64] B. Chen, T. Zhang, H. Geng, C. Zhang, P. Li, K. Song, W. T. Freeman, J. Malik, P. Abbeel, R. Tedrake, et al. Large video planner enables generalizable robot control. *arXiv preprint arXiv:2512.15840*, 2025.

[65] S. Fei, S. Wang, J. Shi, Z. Dai, J. Cai, P. Qian, L. Ji, X. He, S. Zhang, Z. Fei, J. Fu, J. Gong, and X. Qiu. LIBERO-Plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

[66] X. Li, K. Hsu, J. Gu, K. Pertsch, O. Mees, H. R. Walke, C. Fu, I. Lunawat, I. Sieh, S. Kirmani, S. Levine, J. Wu, C. Finn, H. Su, Q. Vuong, and T. Xiao. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

[67] W. Pumacay, I. Singh, J. Duan, R. Krishna, J. Thomason, and D. Fox. THE COLOSSEUM: A benchmark for evaluating generalization for robotic manipulation. In *Robotics: Science and Systems (RSS)*, 2024.

Appendix Contents

A Dataset Comparison 15

B Web-Infra: Browser-Based Infrastructure 16

C TaskGen: Task Generation and Deployment 17

D Data Cleaning and Trajectory Refinement 18

E AXIS Data Rendering 19

F $\pi_{0.5}$ Pretraining from Base 22

G LIBERO Post-training from AXIS-Pretrained Parameters 23

H Real-time Inference and Real-world Rollouts 24

A Dataset Comparison

Table 4: Comparison of representative robot manipulation datasets in terms of collection setting, scale, task coverage, and supervision.

Dataset	Year	Setting	Scale	Task Coverage	Data / Supervision
RoboNet [6]	2019	Real robot	162K trajectories	Diverse manipulation tasks	RGB videos, robot actions, gripper states, and action trajectories
LIBERO [13]	2023	Simulation	6.5K trajectories	130 tasks	RGB video, proprioception, language instructions, and expert demonstrations
BridgeData V2 [8]	2023	Real robot	60K trajectories; 24 environments	13 skills	RGB videos, proprioception, goal images, and language instructions
Open X-Embodiment [11]	2024	Real robot	1M+ episodes; 60 datasets	500+ skills	RGB videos, end-effector poses, language instructions, and action trajectories
DROID [9]	2024	Real robot	76K trajectories; 564 scenes	86 tasks	RGB-D videos, proprioception, teleoperated actions, and language instructions
RoboMIND [10]	2024	Real robot	107K trajectories; 96 objects	279 tasks	RGB-D videos, proprioception, language, teleoperation, and failure demonstrations
PartInstruct [20]	2025	Simulation	513 objects; 1,302 tasks; 10K+ demos	16 part-level task classes	Language, part-level 3D object data, skill sequences, and expert demonstrations
Ours	**2026**	**Web-based simulation**	**50,129 episodes**	**207 tasks**	**Browser-based episodes with scalable user-contributed manipulation data**

Table 4 summarizes representative robot manipulation datasets spanning real-world robotics, simulation environments, and large-scale embodied learning benchmarks. Existing datasets generally emphasize either large-scale real-world data collection (e.g., RoboNet, Open X-Embodiment, DROID, and RoboMIND) or highly controlled simulation benchmarks with expert demonstrations (e.g., LIBERO and PartInstruct). While these datasets have substantially advanced robot learning, their collection pipelines often require specialized robotic hardware, expert operators, or centralized infrastructure, limiting scalability and accessibility.

In contrast, our dataset adopts a web-based simulation paradigm that enables robot manipulation data collection directly through standard web browsers. The framework emphasizes three key properties: scalability, continual growth, and community participation. The absence of physical robots, specialized infrastructure, and expert operators significantly lowers participation barriers and enables large-scale data contribution from a broad user community. New tasks, objects, and interaction scenarios can be incorporated continuously as the community expands. The current release contains 50,129 episodes spanning 207 manipulation tasks, illustrating the potential of community-driven data generation for robotic manipulation. We envision this framework as a complementary direction to existing real-world and simulator-based datasets and as a sustainable pathway toward ever-growing robot manipulation corpora.

B Web-Infra: Browser-Based Infrastructure

AXIS is a cross-simulator, cross-machine infrastructure for scalable robot demonstration collection, trajectory processing, and policy evaluation. The key system decision is to separate interactive data collection from compute-heavy backend processing. Contributors teleoperate robots in a browser-based MuJoCo WebAssembly frontend using commodity devices, while uploaded demonstrations are routed to backend machines for validation, cleaning, replay, rendering, augmentation, and export to downstream policy-learning pipelines. This split lowers the entry cost for data contributors without forcing the full pipeline to inherit the limits of a lightweight browser simulator.

At the pipeline level, AXIS follows six stages: browser-based teleoperation and logging, trajectory upload in a unified format, backend replay and realistic augmentation, data cleaning and curation, model training with sim-to-real evaluation, and real-robot deployment. The pipeline is intentionally asymmetric: latency-sensitive interaction remains on commodity web clients, while throughput-sensitive rendering, replay, model training, and evaluation run on backend machines. Data rendering is performed on 8×RTX 4090 GPUs, while all model training and evaluation are performed on 8×A100 GPUs. This organization lets the same demonstration serve multiple purposes, including task verification, cleaned control supervision, realistic RGB rendering, domain-randomized augmentation, downstream policy learning, and evaluation.

The infrastructure is organized around shared task and trajectory abstractions. At the task level, environments expose common observation, action, reset, step, and success-checking interfaces, so a task collected in MuJoCo-WASM can later be verified, replayed, or rendered in a backend simulator without rewriting the task logic. At the trajectory level, each demonstration stores metadata, time-series signals, and episode-level annotations in a unified format. Metadata include the task name, environment identifier, robot embodiment, simulator version, contributor identifier when available, and timestamps. Time-series fields include actions, robot states, observations, and optional object states or visual observations. Episode-level fields include success labels, segmentation markers, failure modes when available, and links to the task-specific success checker.

Table 5: Functional decomposition of the Web-Infra pipeline.

Stage	**Execution site**	**Role in the pipeline**
Task deployment	Browser client	Loads a parameterized task, robot model, scene objects, and success checker into the MuJoCo-WASM frontend.
Teleoperation	Browser client	Maps keyboard, mouse, joystick, or gamepad commands to end-effector targets and joint-level robot actions.
Logging	Browser client	Buffers time-aligned state-action samples inside the physics loop and serializes successful attempts with task metadata.
Upload and indexing	Backend service	Stores trajectories with versioned metadata so they can be retrieved for validation, replay, rendering, and training.
Cleaning and replay	Backend machines	Filters invalid episodes, smooths and resamples motion, verifies success, and reconstructs trajectories for rendering.
Learning export	Backend machines	Packages cleaned and rendered data for imitation learning and VLA training.

The browser frontend is implemented as an end-to-end teleoperation interface that covers task selection, interactive 3D visualization, camera control, control-state display, and automatic trajectory packaging. The core dashboard acts as a heads-up display over the MuJoCo-WASM and WebGL viewport. A top-level status area reports real-time metrics such as elapsed time and rendering rate, while the control panel groups gripper motion, Cartesian movement, rotation, action buttons, and camera controls so new contributors can quickly understand the available inputs. The interface supports keyboard, mouse, virtual joystick, and gamepad control, making data collection compatible with heterogeneous personal computers rather than specialized lab hardware.

To keep browser control responsive, AXIS uses an orchestrated runtime rather than a monolithic UI loop. MuJoCo physics stepping and Three.js rendering are separated from the main React UI

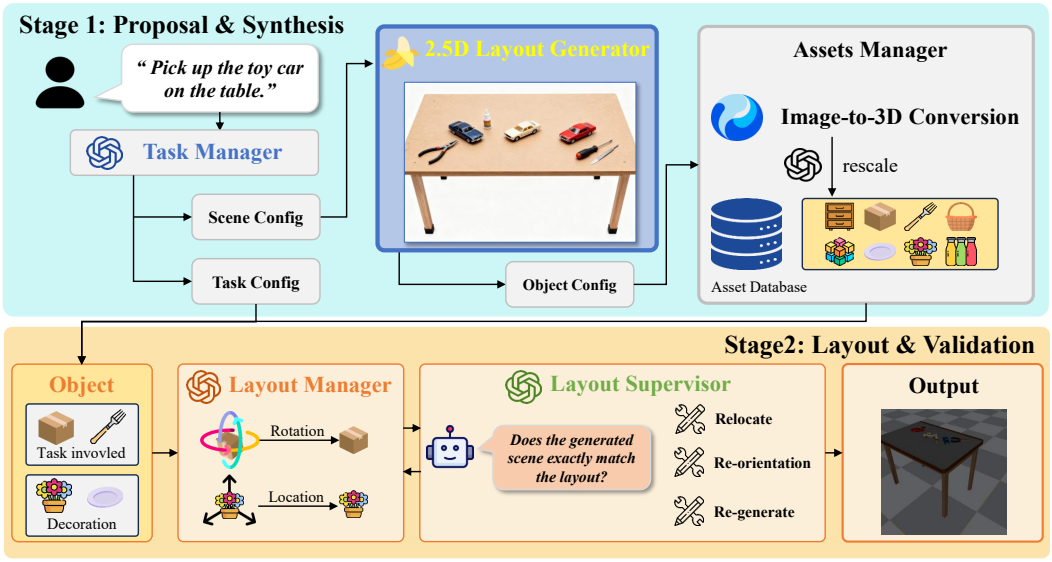

Figure 4: TaskGen pipeline overview. A language instruction is converted into task, scene, and object specifications; object models are retrieved or generated and normalized; a layout is proposed, instantiated, checked, and deployed.

thread. A lightweight broadcaster sends occasional status updates to the interface, while the trajectory manager buffers state-action samples directly inside the high-frequency control loop. This design avoids frequent UI re-rendering during teleoperation and keeps the logged samples aligned with the simulator state. Once the embedded checker verifies task completion, the buffered samples and task metadata are packaged into a standardized trajectory record for backend upload.

Operator commands are mapped to six-degree-of-freedom Cartesian end-effector targets and converted to joint-level robot commands through inverse kinematics. For more complex end-effectors, AXIS also supports interpolation-based grasp control. Let $\mathbf{q}_{\text{rest}} \in \mathbb{R}^n$ denote the fully open hand pose and $\mathbf{q}_{\text{tmpl}} \in \mathbb{R}^n$ denote a grasp template, such as a power, precision, or functional grasp. The commanded hand pose is

$$\mathbf{q}(t) = \mathbf{q}_{\text{rest}} + \alpha(t)\,(\mathbf{q}_{\text{tmpl}} - \mathbf{q}_{\text{rest}})\,, \tag{1}$$

where $\alpha(t) \in [0, 1]$ is controlled by the operator. This collapses a high-dimensional hand configuration, often with more than 16 actuated joints, into a single scalar control axis. New grippers or dexterous hands can therefore be integrated by defining rest poses and task-specific template libraries, without changing the core teleoperation controller.

C TaskGen: Task Generation and Deployment

TaskGen converts a high-level manipulation instruction into a simulation-ready task instance. Given an instruction such as *pick up the toy car on the table*, a task manager decomposes the request into task, scene, and object configurations. The task configuration specifies the manipulation goal and required object interactions. The scene configuration defines workspace context such as supporting surfaces, clutter level, and spatial constraints. The object configuration determines which object models appear in the scene, their semantic roles, and whether they are required for task completion or included as decorative context. Each task is also assigned a difficulty level from 1 to 5, which controls the number of objects, the tightness of spatial constraints, and the overall scene complexity.

TaskGen uses an object-model manager to populate scenes with diverse objects. Object models can be retrieved from an existing database or generated through an image-to-3D conversion pipeline. Because generated meshes may have inconsistent physical scales, the object-model manager normalizes them to plausible dimensions before adding them to the task database. This normalization

step is important for downstream teleoperation and replay: object size affects reachability, grasp feasibility, collision behavior, and the validity of geometric success conditions.

Conditioned on the task specification, selected object models, and difficulty level, a 2.5D layout generator proposes an initial scene arrangement. The generator describes which objects should appear on the workspace and their approximate spatial relationships. A layout manager then instantiates the proposal into a full 3D environment by assigning each object a precise location and orientation. A layout supervisor verifies whether the instantiated scene satisfies the intended task constraints. When the supervisor detects an inconsistency, such as an unreachable object, an invalid support relation, or a mismatch between the language goal and object placement, the system iteratively corrects the scene through relocation, re-orientation, or partial regeneration.

Table 6: TaskGen pipeline components and their outputs.

Component	**Input**	**Output / validation role**
Task manager	Language instruction, difficulty level	Structured task, scene, and object configurations with semantic object roles.
Object-model manager	Object requirements and model database	Retrieved or image-to-3D generated object models normalized to plausible physical scale.
2.5D layout generator	Task specification, object models, clutter settings	Initial object arrangement over the workspace, including task-involved and decorative objects.
Layout manager	2.5D proposal and object geometry	Full 3D scene with object poses, orientations, support relations, and workspace placement.
Layout supervisor	Instantiated scene and task constraints	Validated scene or corrective actions such as relocation, re-orientation, or partial regeneration.
Success checker	Task goal and object interactions	Automatic completion test reused during browser collection and backend verification.

After a valid layout is produced, TaskGen defines distributions over initial conditions to support scenario sampling. These distributions vary object poses, clutter configurations, and camera viewpoints while preserving the core task semantics. Each generated instance is paired with a task-specific success checker that encodes completion using geometric constraints and object interactions. The checker is used both during browser teleoperation, where it terminates and packages successful demonstrations, and during backend validation, where it helps verify whether uploaded trajectories remain task-consistent after transfer.

The output of TaskGen has three parts: a parameterized task definition describing scene configuration and object layout, a unified observation and action interface for execution across simulators, and a task-specific success checker for automatic evaluation. This decomposition makes task growth continuous rather than one-off. New language instructions, objects, layouts, and difficulty settings can be added to the task pool, deployed to the browser, collected by contributors, and then replayed or rendered through the same backend pipeline.

D Data Cleaning and Trajectory Refinement

Raw web demonstrations require filtering and refinement before they can be used for policy learning. Browser teleoperation can introduce jitter, stalled samples, unintended oscillations, low-frequency sampling artifacts, corrupted records, and occasional task-completion mismatches. AXIS therefore treats data cleaning as a production pipeline with two goals: filter unusable episodes and refine borderline episodes into temporally smooth, replayable trajectories.

The validation stage removes trajectories that violate basic structural, numerical, physical, or task-level checks. Structural checks reject demonstrations with missing required fields, incomplete metadata, corrupted records, or length mismatches between state and action streams. Numerical checks reject non-finite values and anomalous records. Physical checks compute successive changes in robot states and reject trajectories whose deltas exceed predefined plausible thresholds, which catches extreme discontinuities and invalid state transitions. Task-level checks remove failed

Algorithm 1 Trajectory cleaning and refinement.

Require: Raw episode S, static threshold ϵ, raw timestep Δt_{raw}, target timestep Δt, Savitzky–Golay parameters W, P

Ensure: Smoothed and resampled episode S_{final}

Step 1: Validation and segmentation

Reject S if required fields, finite values, length consistency, physical continuity, or task success checks fail.

$S_{\text{clean}} \leftarrow \textsc{RemoveStaticSegments}(S, \epsilon)$

$T_{\text{clean}} \leftarrow \textsc{RebuildUniformTime}(S_{\text{clean}}, \Delta t_{\text{raw}})$

$Q_{\text{clean}} \leftarrow \textsc{RobotJointSequence}(S_{\text{clean}})$

Step 2: Robot-motion smoothing

$Q_{\text{smooth}} \leftarrow \textsc{SavitzkyGolay}(Q_{\text{clean}}, W, P)$

▷ Discrete gripper transitions can be excluded from continuous smoothing when needed.

Step 3: Fixed-frequency resampling

$T_{\text{new}} \leftarrow \textsc{Arange}(T_{\text{clean}}[0], T_{\text{clean}}[-1], \Delta t)$

$Q_{\text{final}} \leftarrow \textsc{CubicSpline}(T_{\text{clean}}, Q_{\text{smooth}})(T_{\text{new}})$

Step 4: Extra-state alignment and export

$S_{\text{final}} \leftarrow \textsc{AlignNonRobotState}(T_{\text{new}}, Q_{\text{final}}, S_{\text{clean}})$ **return** S_{final}

episodes and can re-verify nominal success labels using the task-specific checker in the backend simulation environment.

After validation, the pipeline removes uninformative static segments. Samples are classified as static when the absolute variation across all robot joints is below a configured threshold; in the current processing pipeline, this threshold is 5×10^{-3}. Removing these samples eliminates artificial pauses and idle segments caused by human hesitation, while preserving the portions of the episode that contain meaningful control changes. The remaining trajectory is then re-timed to form a consistent temporal sequence for smoothing and resampling.

Continuous robot action or joint trajectories are smoothed with a Savitzky-Golay filter. The current implementation uses a window size of 15 and a polynomial order of 3 for robot motion smoothing, while discrete gripper transitions can be excluded from the smoothing pass so that open-close events do not become physically ambiguous. This stage reduces high-frequency teleoperation artifacts such as abrupt corrections and small oscillations without replacing the original demonstration with an unrelated planned path.

Because browser demonstrations may be recorded at a low or non-uniform rate, cleaned trajectories are resampled to a fixed control frequency with cubic spline interpolation. The web interface can produce trajectories at only 6–8 Hz, while downstream control and policy training require a higher frequency; the current target frequency is 20 Hz. Robot states and continuous action targets are interpolated after smoothing. Non-robot states, such as object poses, bypass the action-smoothing filter and are temporally aligned through interpolation only. The resulting trajectory remains faithful to the demonstrated task execution while becoming smoother, denser, and better aligned with replay and policy-learning requirements.

The cleaned output serves as the handoff point to later rendering and augmentation stages. A retained episode contains validated metadata, temporally aligned robot and object states, smoothed robot motion, success annotations, and episode boundaries. The same cleaned record can be replayed for verification, rendered into RGB observation streams, perturbed through IsaacSim scene or camera randomization, or exported for imitation-learning and VLA training. In this way, data cleaning is not only a quality-control step but also the interface that makes web-collected demonstrations usable by the larger AXIS dataset and model-training pipeline.

E AXIS Data Rendering

AXIS converts verified task trajectories into LIBERO-style visual imitation-learning data through state replay. The rendering source is not an action log that must be re-executed through a simulator. Instead, each eligible attempt is represented as packed simulator state, which provides the robot and

Table 7: Quantitative evaluation of trajectory optimization across 16 LIBERO tasks. Smoothness (Max Acceleration and Max Jerk) is compared before and after the proposed processing.

LIBERO Task	**Smoothness**				**Pos. Dev. (m)**	**Removed Ratio**
	Mean Acc.		**Mean Jerk**			
	Before	After	Before	After		
Task 1: Place Black Bowl on Top of cabinet	0.2458	0.0738	2.6425	1.3363	0.0235	6.46%
Task 2: Place Rear Butter in Cabinet Top Drawer and Close It	0.3067	0.0943	3.7395	1.6208	0.0654	5.73%
Task 3: Place the black bowl on the plate	0.3559	0.1011	5.1358	2.3634	0.0201	7.17%
Task 4: Place the black bowl on top of the cabinet	0.3225	0.0859	4.6724	2.0785	0.0142	3.77%
Task 5: Place the frying pan on the stove	0.1832	0.1055	2.4771	1.1374	0.0106	1.24%
Task 6: Place the moka pot on the stove	0.1819	0.1090	2.5239	1.6784	0.0064	1.94%
Task 7: Turn on the stove	0.2689	0.1295	3.8291	2.3017	0.0142	3.81%
Task 8: Close Cabinet Bottom Drawer	0.1574	0.0741	1.9118	0.6659	0.0057	4.59%
Task 9: Place the black bowl into the cabinet's bottom drawer	0.1551	0.0913	1.8665	1.3193	0.0161	2.03%
Task 10: Place Wine Bottle on Wine Rack	0.1841	0.1114	2.0298	1.0022	0.0625	4.86%
Task 11: Close Cabinet Top Drawer	0.1224	0.0683	1.5424	0.5822	0.0053	2.62%
Task 12: Place the black bowl into the cabinet's top drawer	0.2109	0.1418	2.6997	2.1633	0.0961	11.4%
Task 13: Place Black Bowl on Plate	0.1682	0.0816	2.2262	1.1552	0.2511	2.14%
Task 14: Place the black bowl on top of the cabinet	0.1375	0.0736	1.8499	0.9924	0.0108	6.73%
Task 15: Place the right moka pot on the stove	0.1890	0.1130	2.5768	1.4929	0.0134	3.19%
Task 16: Turn off the stove	0.2149	0.1213	2.8268	1.5823	0.0219	8.93%
Average	**0.2128**	**0.0985**	**2.7844**	**1.4670**	**0.0398**	**4.79%**

object states required for timestep-by-timestep replay. This distinction matters because an action rollout can diverge from the originally verified trajectory under small simulator, contact, or controller differences, while state replay keeps the rendered visual stream aligned with the successful task execution.

Task admission is conservative. A task must have enough verified attempts both remotely and locally, and each selected attempt must be long enough to support action-chunk training. Candidate attempts are ranked by shortest verified simulation time and then by attempt id, which biases the rendered corpus toward compact successful demonstrations. The default setting renders all eligible attempts for an admitted task, so the data-generation stage emphasizes coverage once the verification threshold has been met.

Every rendered episode contains two RGB observation streams: a third-view camera that observes the workspace from outside the robot and a wrist camera mounted near the end-effector. This two-view design is used consistently throughout rendering, AXIS pretraining, and LIBERO post-training.

During rendering, the simulator state is set explicitly at each timestep. The replay record includes robot joint position, joint velocity, joint position target, object position, object rotation, object linear velocity, and object angular velocity. The joint position target is retained because it supports downstream action supervision and consistency checks, while the object state fields are required to reconstruct task-relevant interactions without relying on a fresh dynamics rollout.

In the IsaacSim rendering path, physics stepping is disabled during replay. The replay state is therefore authoritative: the renderer refreshes cameras and sensors after the state is set, but it does not use a new physics integration step to determine the next state. This makes the data-generation process closer to visualizing a verified trajectory than to resimulating a controller. It also explains why table-height or scene-geometry randomization must be coupled with corresponding adjustments

Table 8: Default rendering, camera, and domain-randomization parameters.

Category	Parameter	Value
Rendering backend	Simulator	IsaacSim / IsaacLab
Rendering backend	Render mode	Ray-traced lighting
Rendering backend	Render resolution	256×256
Rendering backend	Camera data type	RGB
Rendering backend	Depth	Disabled by default
Robot	Robot model	Franka, fixed base, position control
Third-view camera	Position	$(1.21, 0.0, 0.885)$
Third-view camera	Look-at target	$(0.3, 0.0, 0.0)$
Third-view camera	Focal length	24.0
Third-view camera	RGB field of view	70° horizontal, 43° vertical
Third-view camera	Focus distance	400.0
Third-view camera	Clipping range	$(0.05, 100000.0)$
Wrist camera	Mount link	Franka hand
Wrist camera	Local position	$(0.16, 0.0, 0.01)$
Wrist camera	Local quaternion	$(0.0, -0.38268343, 0.0, -0.92387953)$
Wrist camera	RGB field of view	87° horizontal, 58° vertical
Third-view camera randomization	Position delta	± 0.10 m along each axis
Third-view camera randomization	Roll/pitch/yaw delta	$\pm 8^\circ$
Wrist camera randomization	Depth / lateral / vertical delta	± 0.02 m, ± 0.015 m, ± 0.005 m
Wrist camera randomization	Roll / pitch / yaw delta	$\pm 8^\circ$, $\pm 4^\circ$, $\pm 4^\circ$
Scene randomization	Randomization level	Level 3
Scene randomization	Scene mode	Mode 3, full USD scene
Lighting randomization	Main light intensity	12,000–35,000
Lighting randomization	Corner light intensity	5,000–15,000
Lighting randomization	Main color temperature	2,800–6,500 K
Lighting randomization	Sphere color temperature	2,500–6,000 K
Lighting randomization	Disk orientation delta	$\pm 15^\circ$
Lighting randomization	Sphere position delta	$\pm(0.5, 0.5, 0.3)$ m

to replayed robot/object states and camera look-at targets. The randomized scene changes the visual and geometric context, but the replayed trajectory must remain physically placed on the randomized table.

Validation is performed at the demonstration level. A demonstration is rejected if required RGB videos or metadata are missing, if joint traces are empty, non-finite, all zero, length-mismatched, or too short for action-chunk training. The validated output contains the third-view RGB stream, wrist RGB stream, metadata, joint-position trace, joint-position-target trace, and episode boundary information. These elements are then packaged into training-ready records with third-view RGB, wrist RGB, state, and 7D LIBERO-style joint-position actions.

The default rendering backend uses IsaacSim / IsaacLab with ray-traced lighting. Depth is disabled by default, so the stable visual data consist of third-view RGB and wrist RGB. The third-view camera approximates a fixed external RGB sensor, while the wrist camera is mounted on the Franka hand and approximates an eye-in-hand sensor. Both camera pose randomization and full scene randomization are enabled.

The randomization settings are broader than simple camera jitter. Scene mode 3 enables full USD scene composition, and level 3 includes scene, material, lighting, and camera variation. The intended effect is to increase visual diversity while keeping the underlying state/action supervision tied to verified trajectories. Because depth is disabled by default, most of the rendered data volume comes from RGB videos, metadata, and wrist RGB observations.

F $\pi_{0.5}$ Pretraining from Base

AXIS pretraining uses verified episodes converted into a LIBERO-style visual imitation-learning format. The default snapshot follows the paper-wide AXIS release scale: 207 tasks and 50,129 episodes. Each episode pairs continuous robot state/action supervision with a third-view camera, a wrist camera, and episode-boundary information.

Table 9: AXIS-Datasets snapshot and default $\pi_{0.5}$ input dimensions.

Dataset / Model Parameter	**Value**
AXIS task count	207
AXIS episode count	50,129
Observation cameras	Third-view RGB and wrist RGB
Image resolution	256×256
State dimension	8D robot state
Action dimension	7D joint-position action
Action horizon	10
Model action dimension after padding	32
Model state dimension after padding	32

Training samples are formed as observation-action chunks within individual episodes. For each selected timestep, the sampler reads the current state and image observations, then constructs a 10-step action chunk from the same episode. If the timestep is close to an episode boundary, the action targets are clipped at the episode end. This prevents action supervision from crossing into the next demonstration and makes the action target semantically consistent with the current observation.

The training representation is intentionally simple: each sample contains robot state, a 10-step action target, third-view RGB, wrist RGB, and the episode boundary needed to keep action chunks within a single demonstration. This representation is then mapped into the LIBERO-compatible model input format.

Pretraining initializes from the official $\pi_{0.5}$ base JAX parameters and performs full-model finetuning. The model uses a PaliGemma Gemma-2B backbone and a Gemma-300M action expert. No LoRA adapter is used: the vision encoder, language backbone, and action expert are all trainable. $\pi_{0.5}$ mode is enabled, the action horizon is 10, and the tokenizer uses a maximum prompt length of 200 tokens.

The input view set is intentionally narrow and matches the rendered dataset: the third-view image provides the external scene view, and the wrist image provides eye-in-hand context. State remains a continuous model input rather than being discretized into the language prompt. This keeps language reserved for task instructions while preserving low-level robot state in the continuous observation pathway.

The transform stack first maps third-view and wrist observations into the LIBERO-style observation layout, standardizes image layout, and applies quantile normalization to state and actions. Images are pad-resized from 256×256 to 224×224. During model loss computation, visual augmentation further applies random crop to 95%, rotation in $[-5^\circ, 5^\circ]$, and brightness/contrast/saturation jitter. Prompts are tokenized with the PaliGemma SentencePiece tokenizer. Finally, state is padded from 8D to 32D and action chunks are padded from 7D to 32D; inference slices the output back to the first seven action dimensions.

For $\pi_{0.5}$, state and action normalization use quantile statistics:

$$x_{\text{norm}} = \frac{x - q_{01}}{q_{99} - q_{01} + 10^{-6}} \times 2.0 - 1.0.$$

The statistics are computed over state and actions. Action statistics use the same 10-step horizon as training, including clipping at episode boundaries. Images are not normalized through these statis-

tics; they are converted from uint8 image values into the model's image range during observation construction.

The pretraining loss is flow matching over action chunks, not one-step behavior cloning. Given normalized and padded actions with shape $[\mathrm{batch}, 10, 32]$, the training step samples Gaussian noise and a continuous timestep, forms a noisy action suffix, and trains the model to predict the velocity target:

$$\epsilon \sim \mathcal{N}(0,1), \qquad t \sim \mathrm{Beta}(1.5,1) \times 0.999 + 0.001,$$
$$x_t = t\epsilon + (1-t)a, \qquad u_t = \epsilon - a.$$

The model receives observation prefix tokens and noisy action suffix tokens, then predicts the action velocity for the final action-horizon positions. The loss is mean squared error between the predicted velocity and u_t, averaged over the action horizon and action dimensions. Since actions are padded to 32 dimensions, the unused dimensions become fixed zero targets unless explicit action-loss weights are introduced. Thus, the model is optimized over the full padded action interface even though only the first seven dimensions are returned at inference time.

Table 10: Default $\pi_{0.5}$ AXIS pretraining optimization settings.

Training Parameter	**Value**
Global batch size	8
Training hardware	8×A100 GPUs
Train steps	100,000
Optimizer	AdamW
AdamW β_1	0.9
AdamW β_2	0.95
AdamW ϵ	10^{-8}
Weight decay	10^{-10}
Gradient clipping	1.0 global norm
Peak learning rate	5×10^{-5}
Warmup steps	10,000
Post-warmup learning rate	Effectively constant at 5×10^{-5}
EMA decay	0.999

Training length is reported in optimization steps rather than epochs, because each sample is an observation-action chunk drawn from within an episode. Checkpoints contain training state, inference-facing parameters, and dataset normalization artifacts. When EMA is enabled, the inference-facing parameters use EMA-smoothed weights and are the intended source for downstream LIBERO post-training initialization.

G LIBERO Post-training from AXIS-Pretrained Parameters

LIBERO post-training starts from AXIS-pretrained $\pi_{0.5}$ inference parameters. Only model parameters are transferred. Optimizer state, training step, and EMA state are initialized for a new LIBERO run. This is different from resuming training: resume is only appropriate when continuing the same LIBERO run, because it restores the full training state rather than using the AXIS checkpoint as an initialization source.

The AXIS-to-LIBERO transfer also keeps dataset normalization separate. The AXIS checkpoint provides weights, but LIBERO post-training uses LIBERO-specific normalization statistics stored with the post-training checkpoint. This prevents the LIBERO policy from using AXIS state/action normalization at serving time. The AXIS-to-LIBERO setup uses the final AXIS full-pretraining parameters as initialization and trains the LIBERO stage for 30,000 steps.

LIBERO post-training uses the official LIBERO LeRobot dataset. The data provide third-view camera images, wrist camera images, 8D continuous policy state, task language prompts, and native LIBERO delta actions. Because LIBERO actions are already delta actions, no additional delta con-

version is applied. This keeps the post-training visual input consistent with the AXIS-pretraining setup.

The post-training input representation remains aligned with $\pi_{0.5}$. The third-view camera and wrist camera provide visual observations, state remains an 8D continuous vector before padding, and the action target is a 10-step chunk. The output transform returns the first seven action dimensions. LIBERO-specific normalization statistics are stored with the post-training checkpoint and are required for downstream policy loading.

Table 11: Default LIBERO post-training data, input, and optimization settings.

LIBERO / Post-training Parameter	**Value**
Dataset	Official LIBERO LeRobot release
Expected episodes	1,693
Image shape	$[256, 256, 3]$
State dimension	8D LIBERO policy state
Action horizon	10
Action semantics	Native LIBERO delta action
Extra delta conversion	Disabled
Active cameras	Third-view camera and wrist camera
Train steps	30,000
Global batch size	64
Training and evaluation hardware	8×A100 GPUs
Optimizer	AdamW
Gradient clipping	1.0 global norm
Peak learning rate	5×10^{-5}
Warmup steps	10,000
Post-warmup learning rate	Effectively constant at 5×10^{-5}
EMA decay	0.999

The default 8×A100 setting uses global batch size 64, which preserves approximately eight samples per GPU. The optimization setup mirrors the pretraining learning-rate and EMA choices while changing the data distribution and initialization.

A LIBERO post-training checkpoint must include inference parameters and LIBERO-specific normalization artifacts. Loading only the model weights is insufficient for downstream policy construction because normalization statistics are required to build the trained policy correctly. The checkpoint therefore stores inference-facing model weights, training state for resuming the same run, and dataset-specific normalization statistics.

H Real-time Inference and Real-world Rollouts

This section supplements the main paper's simulation-based LIBERO-Plus evaluations with qualitative real-world rollouts. We evaluate real-time inference on a Franka robot equipped with a gripper. For the qualitative rollouts, we selected two tasks: grasping and pick-and-place.

During real-time inference, the policy receives RGB observations from a global camera and a wrist-mounted camera, together with the continuous robot state. The policy predicts a short-horizon action chunk, which is converted into executable commands for the Franka controller at deployment time. This setup uses the same types of visual and proprioceptive inputs as the training pipeline, while leaving calibration, timing, and low-level controller execution to the real-robot system.

In addition to real-world keyframes, we also include a qualitative comparison between simulation replay and real-world execution. The simulation row visualizes the replayed task trajectory, while the real-world rows show the corresponding view observations. This comparison is intended to illustrate the observation-level correspondence between the replay pipeline and the real-robot setup. These examples are intended as qualitative evidence of data usability and policy executability.

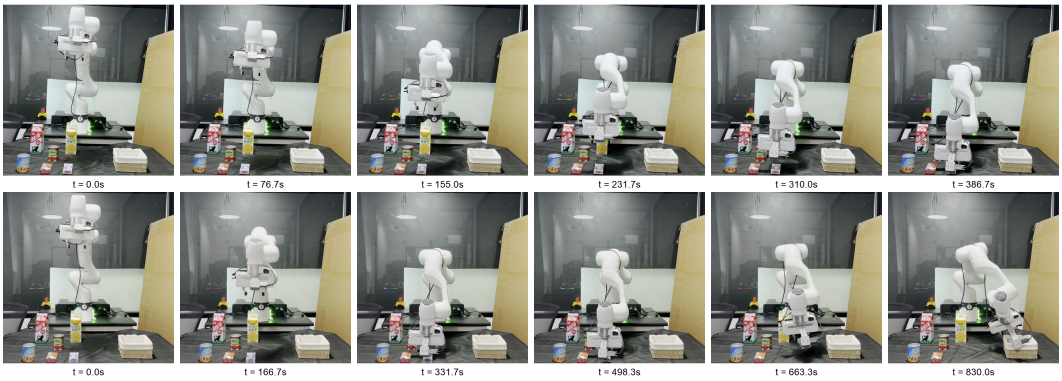

Figure 5: Real-world Franka rollouts. The first row shows a grasping task, and the second row shows a pick-and-place task. Timestamps indicate elapsed real execution time.

Figure 6: Real-time pick-and-place rollout. The two rows show the global and wrist-view.

Figure 7: Pick-and-place replay. The three rows show the simulation replay, real-world global view, and real-world wrist view, respectively.