RoboDojo: A Unified Sim-and-Real Benchmark for Comprehensive Evaluation of Generalist Robot Manipulation Policies

Tianxing Chen[1*§] Yue Chen[4*§†] Zixuan Li[3*] Junyuan Tang[3*] Kailun Su[3*] Haoran Lu[4*] Weijie Wan[3*]
Baijun Chen[1*] Songling Liu[4*] Haowen Yan[3] Honghao Su[3] Zhiyang Dou[6] Kaixuan Wang[1] Dandan Zhang[13]
Yunze Liu[3] Yan Qin[16] Qiwei Liang[16] Qiwei Wu[16] Zijian Lin[3] Wenwei Lin[3] Yuran Wang[10] Minghua He[4]
Tianshu Wu[4] Ruihai Wu[4] Jingquan Zhou[18] Kai-Chong Lei[3] Haibao Yu[1] Yuanfeng Ji[5] Weiyang Jin[1]
Guanyu Lin[9] Xiaofan Li[17] Qi Xiong[3] Renjing Xu[16] Zhongyu Li[12] Wenhao Chai[8] Enze Xie[1] Ziwei Wang[11]
Yao Mu[14] Hao Dong[4] Wojciech Matusik[6] Mingyu Ding[7†] Wenbo Ding[3†] Ping Luo[1†] Masayoshi Tomizuka[2†]

[1]MMLab@HKU [2]UC Berkeley [3]THU [4]PKU [5]Stanford [6]MIT [7]UNC [8]Princeton [9]CMU [10]NUS [11]NTU [12]CUHK [13]IC
[14]SJTU [15]NU [16]HKUST (GZ) [17]ZJU [18]Yale *Co-first Authors §Co-project Leaders †Corresponding Authors

Website: RoboDojo-Benchmark.com Code: Benchmark, XPolicyLab Document Leaderboard

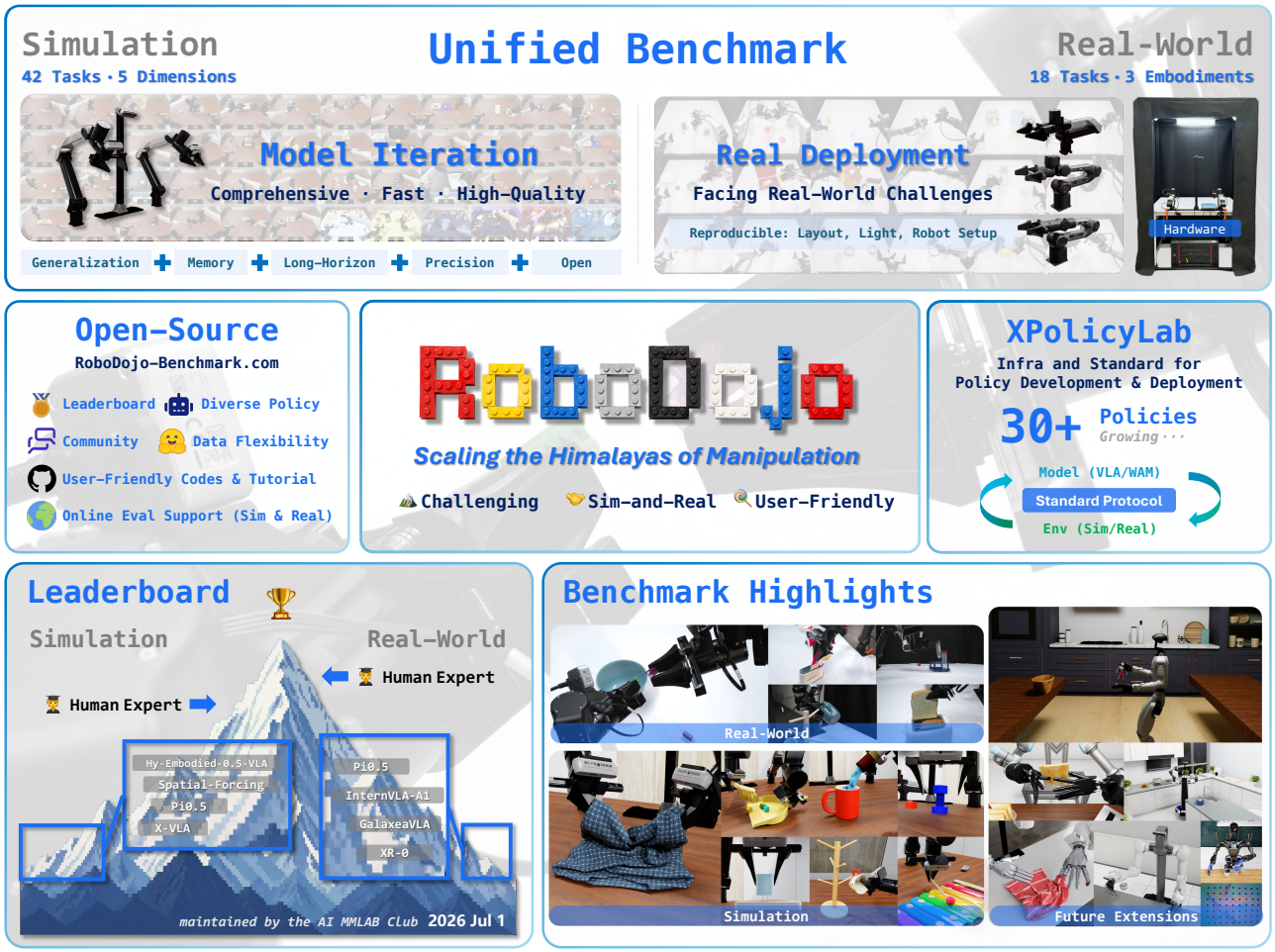


Figure 1: **Overview of RoboDojo.** RoboDojo unifies efficient simulation evaluation and reproducible real-world testing for generalist robot manipulation, covering 42 simulation tasks, 18 real-world tasks, heterogeneous parallel simulation, RoboDojo-RealEval, XPolicyLab, and a continuously updated leaderboard.

Contents

Abstract

Generalist robot manipulation policies have made substantial progress, yet existing benchmarks remain limited in their ability to systematically and comprehensively evaluate policy capabilities. Many benchmarks rely on simple, short-horizon, or skill-narrow tasks that often share similar manipulation patterns and cover only limited capability dimensions. Moreover, evaluations are commonly conducted either in simulation or in the real world alone: simulation provides efficient and scalable feedback but cannot fully capture physical deployment challenges, whereas real-world evaluation offers direct evidence of deployment performance but is costly, time-consuming, and difficult to reproduce. To address these limitations, we introduce **RoboDojo**, a unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies. RoboDojo includes 42 simulation tasks and 18 real-world tasks designed to cover diverse, challenging, and complementary manipulation capabilities. The simulation benchmark evaluates five capability dimensions: generalization, memory, precision, long-horizon execution, and open-vocabulary instruction following, while the real-world benchmark exposes policies to challenging physical-world deployment conditions. To support large-scale evaluation, RoboDojo implements heterogeneous parallel simulation in Isaac Sim, substantially improving evaluation throughput. For real-world evaluation, RoboDojo provides **RoboDojo-RealEval**, a reproducible real-world evaluation system with remote cloud access. By standardizing the hardware setup, scene reset procedure, evaluation protocol, and deployment interface, RoboDojo-RealEval enables policies to be tested under consistent physical conditions. In parallel, **XPolicyLab** provides a unified infrastructure for policy development and deployment, allowing policies to be integrated once and evaluated across RoboDojo simulation and real-world settings with minimal policy-side adaptation. We integrate 30 policies into XPolicyLab and evaluate them on RoboDojo, establishing a public leaderboard with systematic analysis of current policy performance. The website is available at http://robodojo-benchmark.com/.

Leaderboard Governance. To ensure fair and independent evaluation, AI MMLab Club, a non-profit foundation, maintains the RoboDojo leaderboard. Global academic partners will co-govern and operate it without commercial funding or sponsorship.

1. Introduction

A central objective of embodied intelligence is to enable robots to perform manipulation tasks in the physical world. Recent advances in generalist robot policies and embodied foundation models have led to increasingly capable systems across diverse manipulation scenarios (Intelligence et al., 2026b; Zhang et al., 2026c; Li et al., 2026a, 2025b; Zhang et al., 2026a; Lyu et al., 2026; Yuan et al., 2026a). As these systems continue to advance, reliable evaluation becomes essential for understanding their capabilities, limitations, and remaining failure modes. Existing benchmarks have investigated important aspects of robot manipulation, including language-conditioned long-horizon execution (Mees et al., 2022), generalization (Chen et al., 2025a), memory-dependent manipulation (Chen et al., 2026b), precise manipulation (Chen et al., 2026a; Lan et al., 2025), and real-world deployment (Yakefu et al., 2025; Chen et al., 2026c; Atreya et al., 2025). Despite this progress, current evaluation protocols remain insufficient for systematically assessing generalist manipulation policies. Many benchmarks rely on relatively simple or short-horizon tasks, and task variations are often introduced primarily through changes in objects, layouts, or language expressions. While such variations are useful for measuring distributional robustness, they often preserve similar underlying manipulation patterns, limiting diagnostic coverage over distinct challenges such as generalization, memory, precision, open-semantic grounding, sequential execution, bimanual coordination, and tool use.

Another key limitation is the separation between simulation and real-world evaluation. Simulation-based benchmarks are efficient and scalable, making them suitable for rapid feedback and model iteration; however, they cannot fully capture contact-rich dynamics, actuation errors, perception noise, and other physical factors that emerge during deployment. Real-world evaluation provides a more direct assessment of policy behavior under physical-world conditions, but is costly, time-consuming, and difficult to reproduce without standardized hardware setups, scene reset procedures, evaluation protocols, and deployment interfaces. These limitations motivate a unified benchmark that combines scalable simulation-based diagnosis with reproducible real-world validation.

We introduce **RoboDojo**, a unified sim-and-real benchmark for efficient, comprehensive, and reproducible evaluation of generalist robot manipulation policies. RoboDojo contains 42 simulation tasks and 18 real-world tasks. The simulation benchmark provides broad capability coverage across Generalization, Memory, Long-Horizon, Precision, and Open, and supports heterogeneous parallel evaluation in Isaac Sim, enabling different tasks and scenes to run concurrently. The real-world

benchmark complements simulation by exposing policies to challenging physical deployment conditions, including contact-rich interactions, perception noise, actuation errors, and environmental variations. To support reproducible physical testing, we design **RoboDojo-RealEval**, a remote real-world evaluation system that standardizes the hardware setup, workspace layout, lighting condition, scene reset procedure, evaluation protocol, and deployment interface. We further provide **XPolicyLab**, a unified infrastructure for policy development and deployment, enabling policies to be integrated once and evaluated across RoboDojo simulation and real-world settings with minimal policy-side adaptation.

RoboDojo is designed not only to expand benchmark scale, but also to establish a unified evaluation loop for diagnosing and improving generalist robot policies. In simulation, it supports fast policy iteration through diverse task designs and capability-oriented analysis. In the real world, it examines whether policy improvements transfer to challenging physical deployment conditions under standardized and reproducible settings. Together with continuously updated test cases, diagnostic analysis, and a public leaderboard, RoboDojo provides a systematic platform for measuring progress toward robust generalist manipulation. Our contributions are summarized as follows:

- We develop a unified sim-and-real evaluation system for robot manipulation policies, using a shared policy interface and evaluation pipeline across simulation and real-world testing. The system supports heterogeneous parallel simulation for efficient feedback and RoboDojo-RealEval for standardized, reproducible, and remote physical evaluation.
- We construct the **RoboDojo Benchmark**, including 42 simulation tasks and 18 real-world tasks. The simulation tasks cover five capability dimensions: Generalization, Memory, Long-Horizon, Precision, and Open, while the real-world tasks span three robot embodiments and evaluate policies under challenging physical deployment conditions.
- We build **XPolicyLab**, a standardized infrastructure for policy development and deployment. XPolicyLab integrates 30 robot manipulation models into a shared framework, enabling policies to be developed, deployed, and evaluated across RoboDojo benchmarks with minimal policy-side adaptation.
- We conduct extensive evaluations of existing robot manipulation policies on RoboDojo. Based on these results, we establish a public leaderboard and provide systematic analysis of current policy limitations across simulation and real-world settings, highlighting key directions for future generalist manipulation policies.

2. Related Work

2.1. Robot Learning in Manipulation

Robot manipulation policies have rapidly evolved from classical imitation learning methods to large-scale generalist robot policies (Kim et al., 2024; Chen et al., 2025c). Representative imitation learning approaches (Zhao et al., 2023; Chi et al., 2024; Chen et al., 2025b; Ze et al., 2024; Liu et al., 2025b) have achieved strong visuomotor manipulation performance by learning from expert demonstrations. More recent vision-language-action models and foundation policies, including $\pi_{0.5}$ (Intelligence et al., 2025), RDT2 (Team, 2025), JoyAI-RA (Zhang et al., 2026d), Wall-OSS-0.5 (Yu et al., 2026b), Wall-WM (Li et al., 2026b), LDA-1B (Lyu et al., 2026), Motus (Bi et al., 2026a), G0.5 (Galaxea Team, 2026), MotuBrain (Team et al., 2026), Qwen-RobotManip (Yuan et al., 2026a), Hy-Embodied-0.5-VLA (Zhang et al., 2026a) and LingBot-VA (Li et al., 2026a), further improve open-vocabulary instruction following, cross-task generalization, and cross-embodiment transfer. In parallel, recent studies have explored specific capability bottlenecks, such as memory-augmented manipulation (Shi et al., 2025; Torne et al., 2026), dynamic manipulation (Cai et al., 2026b), and world-model-based control (Intelligence et al., 2026a; Bi et al., 2026a; Ye et al., 2026c).

Despite this progress, existing policies still exhibit limited robustness when deployed beyond controlled task distributions. Their failures can arise from diverse factors, including insufficient temporal reasoning, weak spatial precision, limited generalization, and sensitivity to physical-world variations. This motivates evaluation protocols that go beyond aggregate task success and provide structured diagnosis across complementary manipulation capabilities and deployment settings. RoboDojo follows this direction by evaluating generalist policies through both capability-oriented simulation tasks and standardized real-world tests.

2.2. Evaluation for Robot Manipulation

A wide range of benchmarks have been proposed for evaluating robot manipulation. Simulation benchmarks provide scalable and reproducible testbeds: CALVIN (Mees et al., 2022) focuses on language-conditioned long-horizon manipulation;

LIBERO (Liu et al., 2023) studies lifelong learning across spatial, object, goal, and long-horizon suites; RoboTwin (Chen et al., 2025a) evaluates bimanual manipulation and generalization under structured domain randomization; RMBench (Chen et al., 2026b) targets memory-dependent manipulation; GarmentLab (Lu et al., 2024) focuses on garment manipulation; and SimplerEnv (Li et al., 2024a) studies simulation environments that correlate with real-world performance. Real-world benchmarks, including ManipulationNet (Chen et al., 2026c), RoboChallenge (Yakefu et al., 2025), and RoboArena (Atreya et al., 2025), further assess policy behavior under physical deployment conditions.

These benchmarks have substantially advanced robot policy evaluation, but they are often specialized to particular task families, capability axes, or evaluation environments. Simulation-only benchmarks enable efficient and controlled stress testing, yet cannot fully capture the physical uncertainties of real-world deployment. Real-world benchmarks provide more direct evidence of deployed performance, but reproducibility remains difficult when hardware setups, scene reset procedures, evaluation protocols, and deployment interfaces vary across users or sites. RoboDojo addresses this gap by providing a unified sim-and-real benchmark with a shared policy interface and evaluation pipeline. It combines capability-oriented simulation diagnosis with reproducible real-world validation, while supporting policy integration through XPolicyLab. A detailed comparison between RoboDojo and existing benchmarks is provided in Appendix B.

3. RoboDojo Benchmark

Building on the system design in Section 4, we construct the **RoboDojo Benchmark** to evaluate generalist robot manipulation policies across simulation and the real world. RoboDojo consists of 42 simulation tasks and 18 real-world tasks. The simulation benchmark supports efficient model iteration and capability-oriented diagnosis across five dimensions: Generalization, Memory, Precision, Long-Horizon, and Open. These dimensions capture distinct manipulation requirements beyond variations in objects, layouts, or language instructions. The real-world benchmark complements simulation by evaluating policies under challenging physical deployment conditions with standardized and reproducible protocols. Together, RoboDojo provides a broad evaluation suite for systematically diagnosing the limitations of current embodied manipulation policies.

3.1. Simulation Benchmark

We design 42 simulation tasks on the ARX X5 bimanual platform, with the two arm bases separated by 0.6,m. These tasks are organized into five capability dimensions: Generalization, Memory, Long-Horizon, Precision, and Open. The simulation benchmark is designed to provide rapid policy feedback and capability-oriented diagnosis beyond repeated variations of a narrow skill set. Together with the heterogeneous parallel simulation described in Section 4.1.3, RoboDojo enables efficient model iteration across diverse task structures and manipulation requirements. An overview of all simulation tasks is shown in Fig. 2. Detailed task definitions and visualizations are provided in Appendix I and the project documentation.

3.1.1. Task Design

Generalization. Robust deployment requires policies to complete the same task across diverse environments, rather than relying on fixed scenes or familiar visual contexts. The Generalization dimension includes 12 tasks that evaluate robustness to unseen variations in backgrounds, lighting, clutter, and target objects. Compared with RoboTwin 2.0 (Chen et al., 2025a), RoboDojo introduces stronger scene randomization and denser tabletop clutter. While RoboTwin 2.0 includes at most 10 clutter objects, RoboDojo randomizes up to 25 clutter objects, substantially increasing visual distraction and scene complexity. To further reduce overfitting to the default wooden-table scene, we provide 100 supplementary DLC trajectories for data-level augmentation. These trajectories are collected from simple pick-and-place behaviors under randomized backgrounds, lighting, and clutter layouts. Since they are independent of the evaluation tasks, they improve visual exposure without directly leaking task-specific solutions.

Memory. Many manipulation tasks are partially observable, where the correct action depends on information observed earlier rather than only the current frame. Inspired by RMBench (Chen et al., 2026b), the Memory dimension includes 6 tasks that evaluate long-context observation modeling, key-frame memory, and non-Markovian decision making. These tasks cover both short-history settings, where a few key observations are sufficient for task completion, and longer interactive settings that require reasoning over multi-frame temporal sequences. For example, in `match_and_pick_from_conveyor`, the policy must remember the category of an object that disappears on the conveyor and later pick a matching object from subsequent candidates. In `imitate_sorting_sequence`, the policy observes another robot arm placing objects into a basket in a specific order, and must remember and reproduce the demonstrated ordering. These settings challenge policies that rely only on the current observation or a short fixed-length history.

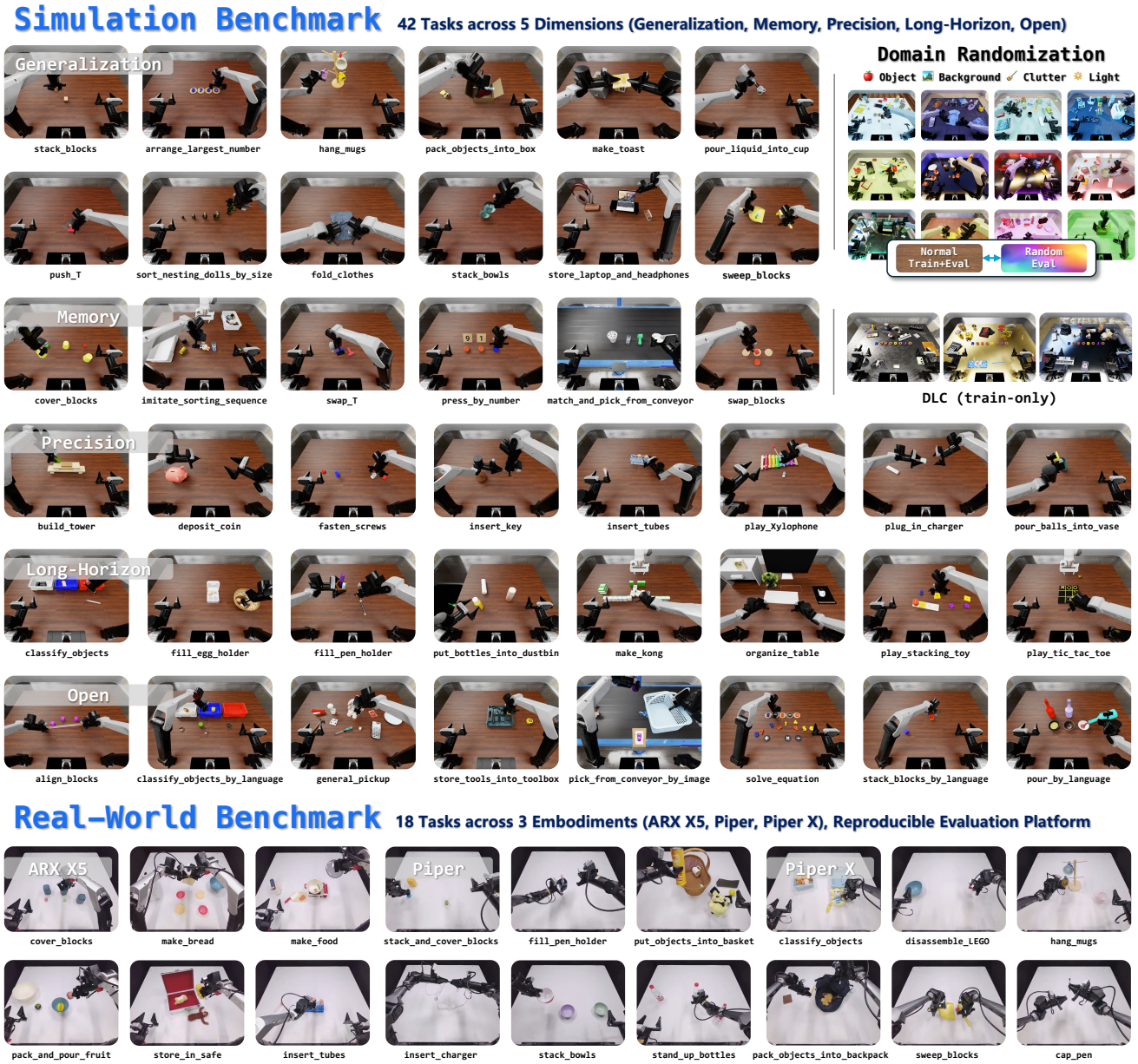


Figure 2: **Task Overview of RoboDojo.** RoboDojo includes 42 simulation tasks and 18 real-world tasks for evaluating generalist robot manipulation policies. The simulation tasks are organized into five capability dimensions: Generalization, Memory, Long-Horizon, Precision, and Open, enabling efficient capability-oriented diagnosis. The real-world tasks assess policy behavior under challenging and reproducible physical deployment conditions. Together, these tasks cover diverse manipulation skills, spatial configurations, and bimanual coordination patterns across simulation and the real world.

Long-Horizon. Practical manipulation often requires executing a sequence of dependent steps before reaching the final goal. The Long-Horizon dimension includes 8 tasks that evaluate whether policies can infer task structure, maintain progress, and complete all required sub-steps without early termination or accumulated errors. For example, `classify_objects` requires the robot to sort multiple objects into corresponding target locations. Due to the relatively large workspace, the task may also require handover behaviors between the two arms, resulting in longer execution horizons and increased temporal complexity.

Precision. Fine-grained manipulation requires accurate visual grounding, smooth motion prediction, and stable local control, since small spatial errors can directly lead to failure. The Precision dimension includes 8 tasks with strict spatial and motion accuracy requirements, focusing on fine-grained target localization, trajectory smoothness, and contact-rich control stability. For example, in `insert_tubes`, the policy must continuously insert tubes into narrow holes. Even small localization or

trajectory errors can prevent successful insertion, making this dimension particularly sensitive to action accuracy and motion stability.

Open. Generalist robot policies should understand diverse task specifications and transfer learned skills to new goals. The Open dimension includes 8 tasks that evaluate unseen task specifications whose required skills appear in the training data under different contexts. These tasks test open-semantic grounding, skill recombination, and language-conditioned transfer. For example, `general_pickup` requires object grasping, a skill frequently observed during training, while the specific grasping goal is not directly included in the training tasks. This dimension examines whether policies can recombine learned manipulation skills and ground new language instructions to solve previously unseen tasks.

3.1.2. Training Data Setting

For simulation policy training, we provide 35 task directories with 3,500 trajectories, totaling 1,859,602 frames and 20.66 hours of bimanual manipulation data recorded at 25 Hz. These directories include 34 training tasks from the Generalization, Memory, Long-Horizon, and Precision dimensions, together with one auxiliary DLC data directory. Each training task contains 100 trajectories collected through either automated trajectory synthesis or VR-based teleoperation. The Open dimension is excluded from the training set, as it is designed to evaluate skill recombination and transfer to unseen task specifications rather than imitation of task-specific demonstrations.

For the Generalization dimension, demonstrations are collected under the standard setting without domain randomization. To reduce overfitting to a single visual configuration, we additionally include 100 auxiliary DLC trajectories for data-level augmentation. These trajectories are generated under complex domain randomization, including diverse backgrounds, lighting conditions, and clutter layouts. The auxiliary DLC data broadens visual exposure while remaining independent of the evaluation tasks, avoiding direct leakage of task-specific solutions. Detailed data modalities, collection procedures, and statistics are provided in Appendix D.1.

3.1.3. Evaluation Setting

We evaluate policies on all 42 simulation tasks across five capability dimensions: Generalization, Memory, Long-Horizon, Precision, and Open. Each task is evaluated for 50 episodes, resulting in 2,100 episodes in total. We report both success rate and average score: success rate measures binary task completion, while average score captures partial task progress.

For the 12 Generalization tasks, the 50 episodes are split into 25 *standard* episodes and 25 *random* episodes. The *standard* setting evaluates in-domain performance, whereas the *random* setting introduces variations in backgrounds, clutter layouts, task-relevant objects, and lighting conditions to assess robustness under visual and scene-level distribution shifts. The overall performance is computed as the mean across the five capability dimensions, rather than across all tasks, preventing dimensions with more tasks from dominating the aggregate metric. Additional details on task horizons and episode settings are provided in Appendix D.2.

3.2. Real-World Benchmark

To evaluate policy performance under physical deployment conditions, we construct the **RoboDojo Real-World Benchmark**. The benchmark consists of 18 real-world tasks across three commonly used collaborative bimanual robot platforms: ARX X5, Piper, and Piper X. This multi-embodiment design evaluates whether policies can maintain reliable performance across different robot kinematics, workspaces, camera placements, and data distributions, rather than being specialized to a single platform.

Unlike simulation tasks, which support rapid feedback and capability-oriented diagnosis, the real-world benchmark exposes policies to deployment challenges that are difficult to fully capture in simulation, including contact-rich interactions, perception noise, actuation errors, workspace constraints, and temporal variations. As described in Section 4.2, we open-source the hardware setup and evaluation specifications of **RoboDojo-RealEval** to support reproducible deployment. RoboDojo-RealEval standardizes the hardware configuration, workspace layout, lighting condition, scene reset procedure, evaluation protocol, and deployment interface, enabling policies to be tested under consistent physical conditions. We further provide cloud-based remote evaluation, allowing users to submit policies and evaluate them under shared physical test configurations. Detailed task information is provided in Appendix J and the project documentation.

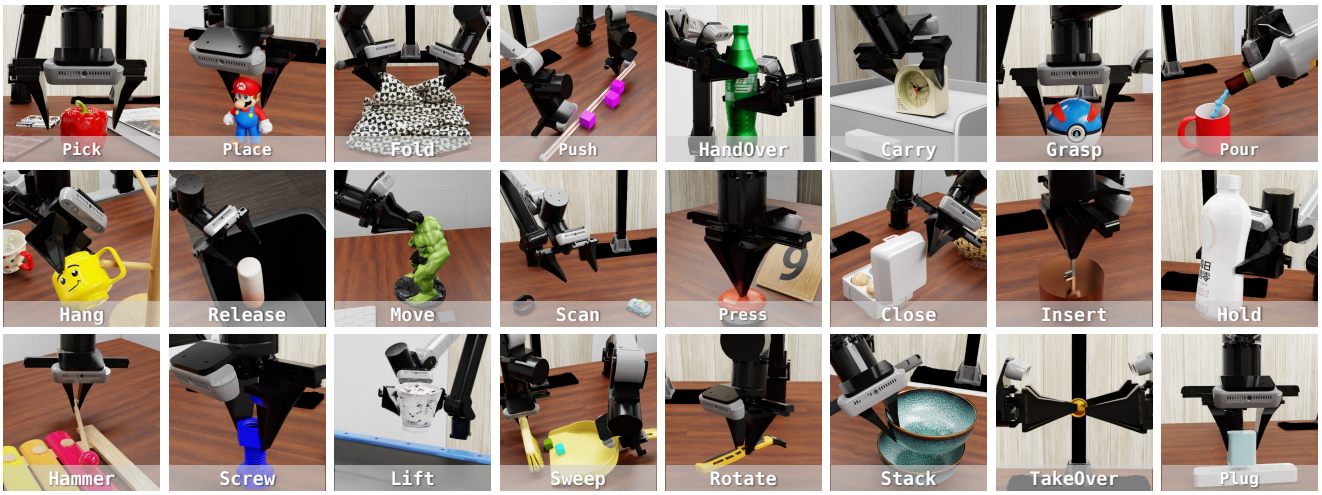


Figure 3: **Skill Diversity in RoboDojo.** Representative simulation tasks cover 24 manipulation skills, including grasping, placing, pushing, pulling, stacking, insertion, opening, closing, folding, alignment, tool use, and contact-sensitive operations. These tasks span diverse spatial configurations and bimanual coordination patterns, enabling skill-level diagnosis beyond repetitive pick-and-place behaviors.

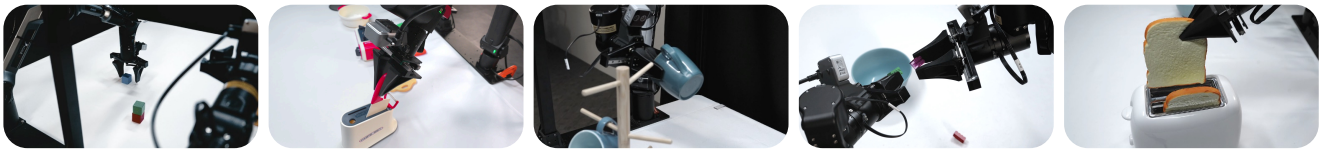


Figure 4: **Real-World Task Highlights.** Representative key frames of challenging real-world manipulation tasks across different robot embodiments.

3.2.1. Task Design

We design 6 tasks for each embodiment, resulting in 18 real-world tasks in total. Since real-world evaluation is constrained by hardware cost, execution time, human reset effort, and safety considerations, the benchmark is designed as a compact yet challenging physical test suite rather than an exhaustive enumeration of all capability dimensions. The selected tasks cover representative requirements, including long-horizon execution, precise manipulation, memory, and generalization.

Importantly, we do not construct one-to-one aligned task pairs between simulation and the real world. The real-world benchmark is not intended to measure direct sim-to-real transfer on matched tasks. Instead, it serves as a complementary evaluation setting for testing whether policies can handle physical deployment challenges beyond simulation success. The tasks involve contact-rich interactions, fine-grained spatial alignment, multi-object manipulation, partial observability, and embodiment-specific motion constraints, making evaluation sensitive to perception errors, actuation noise, accumulated trajectory drift, and unstable contact dynamics.

By evaluating policies across ARX X5, Piper, and Piper X under a shared protocol, RoboDojo assesses whether current generalist manipulation policies can operate reliably across diverse real-world deployment conditions. Detailed task specifications are provided in Appendix J.

3.2.2. Training Data Setting

We collect real-world demonstrations using a homogeneous leader-follower teleoperation setup, where the leader arm has the same embodiment as the follower robot. This design aligns the teleoperation command space with the deployed robot embodiment, providing stable demonstrations for policy training. For each task, we collect 100 demonstrations from four operators to increase behavior diversity and reduce operator-specific bias. Across three embodiments and 18 tasks, the dataset contains 1,800 trajectories and 1,611,841 frames, corresponding to 17.91 hours of bimanual manipulation data recorded at 25 Hz.

Each demonstration includes robot states, language annotations, and synchronized RGB observations from one head camera and two wrist cameras. After collection, all trajectories are manually inspected for task completion, demonstration quality, and consistency with the RoboDojo-RealEval reset protocol. Detailed data modalities, filtering criteria, and embodiment-wise statistics are provided in Appendix E.1.

3.2.3. Evaluation Setting

For real-world evaluation, we use the RoboDojo-RealEval platform introduced in Section 4.2. RoboDojo-RealEval supports both local deployment and remote cloud-based evaluation, where policies communicate with the real-robot client through the protocol described in Section H. For each task, each policy is evaluated over 10 trials. Before each trial, a pre-collected evaluation layout is replayed to ensure consistent initial conditions across policies and sessions.

All trials are recorded and independently scored by three evaluators under a double-blind protocol. The scoring accounts for both final task success and intermediate sub-step completion, enabling fine-grained assessment of partial progress. The final score of each trial is computed as the average of the three evaluator scores. To improve transparency and fairness, we release evaluation videos and scores for leaderboard submissions, and provide an appeal mechanism for potential scoring errors. Additional details on motion planning, execution horizons, safety termination, and scoring protocols are provided in Appendix E.2.

3.3. Evaluation Integrity and Anti-Gaming Protocols

Leaderboard Governance. The RoboDojo leaderboard is maintained by AI MMLab Club, a non-profit foundation dedicated to supporting open and community-driven AI research. To ensure fairness, neutrality, and long-term credibility, RoboDojo evaluation will be jointly maintained and operated by global academic institutional partners. No commercial company, including through sponsorship, funding, or operational participation, is involved in the official evaluation process. As the benchmark evolves, the official evaluation and publication rules may be updated accordingly. The latest leaderboard rules and submission protocol are available at http://RoboDojo-Benchmark.com/Leaderboard.

Although RoboDojo open-sources the simulation benchmark and the structural design of the real-world evaluation platform, official leaderboard publication requires a standardized verification protocol to ensure fairness, reproducibility, and community supervision. RoboDojo uses released public layouts as the primary leaderboard evaluation settings, allowing users to inspect task layouts, reproduce evaluations, and compare methods under a shared protocol. To reduce overfitting, hand-tuning, and leaderboard gaming on public layouts, submitted models are additionally evaluated on hidden verification layouts during official evaluation. These hidden layouts serve as an auxiliary consistency check rather than the primary leaderboard test set.

RoboDojo supports remote evaluation without requiring participants to release checkpoints or source code during private development. Participants can run a remote policy server and communicate with the official RoboDojo evaluation client through the standardized deployment protocol. To publish results on the official verified leaderboard, each submission must satisfy the following requirements: **(1)** policies are evaluated through the official RoboDojo online evaluation system; **(2)** simulation results are reported over three random seeds with mean and standard deviation, and real-world results cover all three robot embodiments, including ARX X5, Piper, and Piper X; **(3)** submitted models pass hidden-layout verification; **(4)** the evaluated checkpoint, training and deployment code, configuration files, and reproducibility instructions are released through **XPolicyLab** at the leaderboard publication stage; and **(5)** evaluation videos are released for community inspection.

Figure 5: RoboDojo official evaluation is jointly supported by global academic institutional partners, with no commercial participation.

Participants may still use RoboDojo for private evaluation and iteration without releasing code or checkpoints. However, results without the evaluated checkpoint, implementation, configuration files, and reproducibility instructions are reported separately and are not considered verified leaderboard entries. In this way, RoboDojo balances flexible remote development with a verified publication protocol that emphasizes reproducible policy performance rather than hand-tuned results on specific released layouts. Additional details of the verification and publication protocol are provided in Appendix A.2.

4. Technical Implementation

In this section, we present the technical design of the **RoboDojo** evaluation platform. RoboDojo comprises three main components: a simulation evaluation platform, a real-world evaluation platform, and **XPolicyLab**, a unified infrastructure for policy development and deployment. The simulation platform supports diverse task construction, digital-twin asset generation, heterogeneous parallel evaluation, and simulation data collection, enabling rapid feedback and efficient policy iteration across manipulation capabilities. The real-world platform, **RoboDojo-RealEval**, enables reproducible physical evaluation by standardizing the hardware configuration, workspace layout, lighting condition, scene reset procedure, evaluation protocol, and deployment interface. XPolicyLab defines a shared policy interface for observation updates, action prediction, policy reset, and batched policy queries, allowing different robot manipulation policies to be integrated into a common codebase and evaluated under a unified protocol. Together, these components enable policies to be integrated once, iterated efficiently in simulation, and deployed to standardized real-world evaluation with minimal policy-side adaptation.

4.1. Simulation Platform

To support efficient and reproducible evaluation of generalist manipulation policies, RoboDojo builds a configurable simulation platform based on NVIDIA Isaac Sim (NVIDIA, 2025) and Isaac Lab (Mittal et al., 2025). The platform comprises three main components: a configuration-driven simulation stack, a physically grounded digital asset library, and a heterogeneous parallel execution mechanism. Together, these components support diverse task construction, scalable policy evaluation, and efficient demonstration collection. Additional implementation details are provided in Appendix F.1.

4.1.1. Simulation Platform Setup

RoboDojo uses the vectorized execution interface of Isaac Lab (Mittal et al., 2025) as the simulation backend, and builds its simulation stack on the infrastructure of MagicSim (Lu et al., 2026), inheriting its modular manager architecture, configuration-driven scene construction pipeline, and object simulation backend. Each task is instantiated from modular YAML specifications that define task assets, scene layouts, initialization distributions, randomization ranges, and success conditions. This design decouples task specification from simulator execution, allowing different tasks to share the same runtime while varying task-specific objects, scene layouts, and reset distributions, details can be found in F.

At reset time, RoboDojo samples task-relevant objects, object poses, articulation states, clutter layouts, lighting conditions, and background textures with deterministic seed control. This enables diverse yet reproducible scene instances for both training and evaluation. The same scene construction pipeline supports rigid, articulated, and deformable assets through a unified interface. Details of the configuration format, randomization parameters, and scene construction pipeline are provided in Appendix F.2.

4.1.2. Physically Grounded Digital Asset Library

RoboDojo builds a digital asset library containing rigid, articulated, and deformable objects. These assets are used as both task-relevant objects and clutter distractors across simulation tasks, enabling diverse scene instantiations while maintaining reliable physical behavior for contact-rich manipulation, inspired by ManiTwin (Wang et al., 2026b). Rigid objects are collected from online repositories and reconstructed from real-world daily objects using Meshy AI[1]. Each asset is annotated with semantic and task-level metadata, including object categories, language descriptions, placement regions, success-checking annotations, and manipulation affordances.

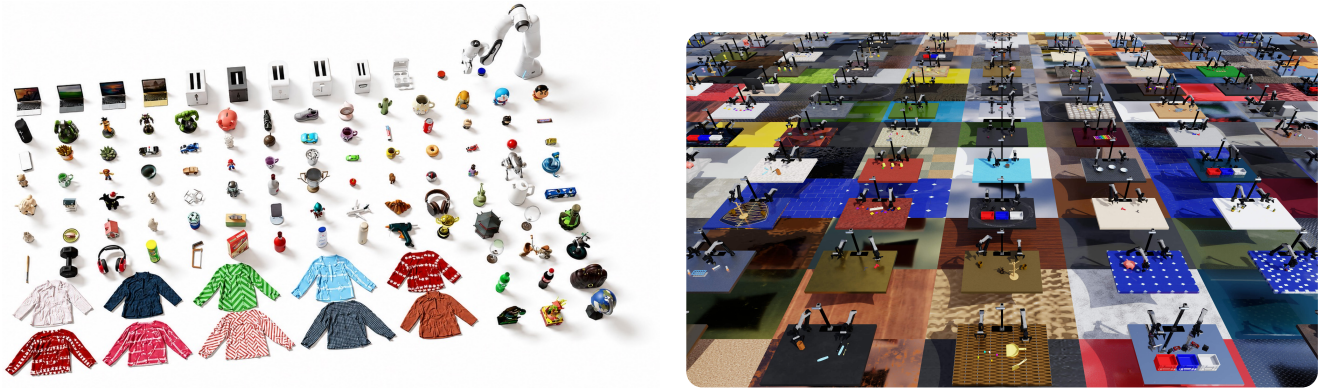
Articulated and deformable assets are collected from open-source platforms, 3D scanning, and selected assets from Clothes-Net (Zhou et al., 2023). Before being used in RoboDojo tasks, these assets are standardized in terms of kinematic structure, collision geometry, material parameters, and simulation-ready representations. Details on asset processing, annotation, and physical validation are provided in Appendix F.3.

4.1.3. Heterogeneous Parallelism

Efficient benchmark evaluation requires parallel simulation over diverse scenes. Standard vectorized simulation often relies on homogeneous cloned environments, where parallel instances share the same scene template and differ mainly in poses or random seeds. While efficient, this setting is insufficient for evaluating generalist manipulation policies, which should be tested across diverse object geometries, object counts, clutter layouts, articulated structures, and background conditions.

RoboDojo therefore supports heterogeneous parallel simulation. Multiple environments are stepped under a shared vectorized

[1] https://www.meshy.ai/



(a) **Asset library.** Sample rigid, articulated, and deformable assets. (b) **Parallel simulation.** Concurrent heterogeneous task execution.

Figure 6: **Assets and parallelism in RoboDojo.** RoboDojo combines physically grounded assets with heterogeneous parallel simulation for scalable benchmark construction and evaluation.

interface, while each environment maintains an independently sampled scene configuration. Different parallel environments can contain different object categories, asset geometries, numbers of distractors, articulation structures, and task layouts. This design improves evaluation speed while preserving the scene-level diversity required for robust policy assessment. Implementation details of heterogeneous parallelism and multi-GPU sharding are provided in Appendix F.4.

4.1.4. Simulation Data Collection

RoboDojo supports two complementary data collection modes: automated trajectory synthesis and VR-based teleoperation. Both modes use a shared asset annotation layer that specifies manipulation-related affordances and task semantics, including graspable regions, placement regions, functional parts, and task-specific interaction points. These annotations support automated skill grounding and task validation.

For automated synthesis, RoboDojo composes demonstrations from reusable low-level skills, such as `grasp`, `place`, `handover`, `insert`, `open`, `close`, `stack`, and `push_up`. Each skill is grounded in asset annotations and executed with the cuRobo v2 motion planner (Sundaralingam et al., 2026). For tasks that are difficult to synthesize automatically, RoboDojo provides a VR-based teleoperation interface for collecting high-quality demonstrations. Details on skill composition, motion planning, and teleoperation control are provided in Appendix F.5.

4.2. Real-World Platform: RoboDojo-RealEval

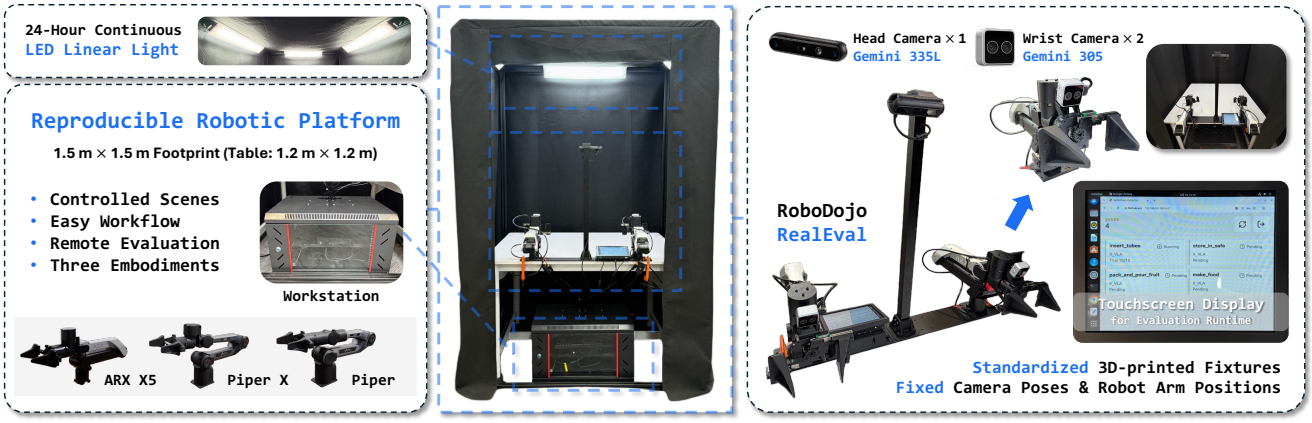


Figure 7: **Overview of the RoboDojo-RealEval system.** RoboDojo-RealEval provides a standardized physical platform for reproducible real-world robot manipulation evaluation, with controlled workspace geometry, fixed robot and camera mounts, stable lighting, a touchscreen evaluation interface, and support for three collaborative bimanual embodiments.

Real-world robot evaluation is highly sensitive to incidental factors such as lighting, robot placement, camera pose, manipulation surface condition, and scene reset accuracy, making reproducibility and fairness difficult to ensure across sessions and sites. To address this issue, we develop the **RoboDojo-RealEval Platform**, shown in Fig. 7, as a standardized hardware and software system for reproducible real-world evaluation. RoboDojo-RealEval fixes the relative poses of robot arms, cameras, lighting units, and the manipulation workspace through modular structural components, and provides a touchscreen-based web interface for scene reset, policy execution, emergency stop, video collection, and cloud-based scoring. During evaluation, predefined task layouts are replayed through a transparent overlay between the target layout image and the live observation stream, enabling evaluators to restore consistent initial conditions before each trial. The platform supports three collaborative bimanual embodiments, including ARX X5, Piper, and Piper X, and enables both local deployment and remote cloud-based evaluation under shared physical protocols. Together, these designs reduce uncontrolled variations and provide a reproducible physical foundation for standardized real-world policy evaluation. Additional hardware details, layout replay procedures, safety controls, and scoring protocols are provided in Appendix G.

4.3. XPolicyLab

XPolicyLab is a unified infrastructure and standard for robot policy development, training, evaluation, and deployment. It is designed to reduce the engineering burden of integrating heterogeneous robot policies into a shared benchmark ecosystem. Existing policies often rely on different data formats, preprocessing pipelines, training scripts, action representations, and runtime environments. XPolicyLab standardizes these external workflows through common data conversion tools, training templates, deployment procedures, and evaluation scripts, while preserving each policy's internal architecture and implementation.

Through XPolicyLab, we integrate 30 representative robot policy models into a shared codebase, enabling consistent training, debugging, deployment, and comparison under the RoboDojo protocol. For evaluation, XPolicyLab connects policy servers with RoboDojo simulation and RoboDojo-RealEval through a standardized observation-action interface. The same policy implementation can therefore be trained and iterated in simulation with fast feedback, and then deployed to remote real-world evaluation with minimal policy-side adaptation. The unified policy interface, communication protocol, and implementation details are provided in Appendix H. The supported policies and project codebase are available at https://XPolicyLab.github.io.

5. RoboDojo Leaderboard

We establish RoboDojo leaderboards to provide standardized comparisons of current robot manipulation policies across both simulation and real-world settings. The simulation leaderboard evaluates broad capability coverage and supports efficient diagnosis through large-scale, repeated trials across five capability dimensions. The real-world leaderboard complements simulation by assessing policy behavior under standardized physical deployment conditions across multiple robot embodiments. Together, these leaderboards provide a unified view of policy performance, revealing both capability-level limitations in simulation and deployment-level challenges in the physical world.

5.1. Simulation Benchmark Leaderboard

Using **XPolicyLab**, we integrate, train, and evaluate 30 representative robot manipulation policies on the RoboDojo Simulation Benchmark Leaderboard. As a human-level reference, we additionally invite three expert teleoperators to perform VR-based simulation teleoperation under the same success criteria and execution horizons used for policy evaluation. Each expert has more than 1,000 hours of simulation teleoperation experience and participated in constructing the RoboDojo Simulation Benchmark dataset.

For most policies, we train three random seeds and evaluate each seed for 50 trials per task. For the Generalization dimension, the 50 trials are split into 25 trials under the standard setting and 25 trials under the randomized setting. Unless otherwise specified, each policy is therefore evaluated over 150 trials per task across three seeds. We report the mean success rate and mean score for each capability dimension, together with their corresponding standard deviations. The overall success rate and score are computed by averaging the corresponding metrics across the five capability dimensions.

The leaderboard is frozen for this paper on *3 Jul. 2026*, while the latest results will be continuously updated at http://RoboDojo-Benchmark.com/Leaderboard. Detailed training configurations for all evaluated models are provided in Appendix K. Models not trained with three random seeds are specially marked in the appendix.

Table 1: **RoboDojo Simulation Benchmark Leaderboard**. Each cell reports score / success rate for a capability dimension. For each metric, rankings among evaluated policies are marked as **Best** and **Second Best**. Human teleoperation is reported as a reference and is excluded from policy ranking. (3 Jul. 2026, continuously updated at RoboDojo-Benchmark.com/Leaderboard.)

Alg \ Dim	Generalization	Precision	Long-Horizon	Memory	Open	*Average*
Hy-Embodied-0.5-VLA (Zhang et al., 2026a)	11.77 / **8.39%**	13.81 / 8.00%	**25.74** / **14.92%**	**13.37** / **12.11%**	0.65 / 0.58%	**13.07** / **8.80%**
Spatial Forcing (Li et al., 2025a)	**14.12** / **9.33%**	**17.33** / **10.58%**	23.26 / 14.58%	5.43 / 4.11%	**1.78** / **1.58%**	**12.38** / **8.04%**
$\pi_{0.5}$ (Intelligence et al., 2025)	**13.37** / 8.17%	12.40 / 5.50%	**23.54** / **14.67%**	5.78 / 4.56%	**1.98** / **1.67%**	11.41 / 6.91%
X-VLA (Zheng et al., 2025)	10.48 / 6.78%	**18.32** / **12.00%**	16.53 / 9.75%	4.76 / 3.56%	0.55 / 0.50%	10.13 / 6.52%
X-WAM (Guo et al., 2026)	7.39 / 3.33%	6.72 / 1.83%	17.47 / 9.08%	**6.32** / 4.67%	0.57 / 0.25%	7.69 / 3.83%
Xiaomi-Robotics-0 (Cai et al., 2026c)	7.43 / 5.56%	8.42 / 4.58%	13.51 / 6.92%	5.07 / 3.67%	0.22 / 0.17%	6.93 / 4.18%
StarVLA-α (Ye et al., 2026b)	3.93 / 2.33%	9.90 / 4.33%	14.15 / 6.50%	3.34 / 2.44%	0.68 / 0.58%	6.40 / 3.24%
GigaWorld-Policy (Ye et al., 2026a)	5.34 / 2.89%	6.15 / 1.83%	15.51 / 8.92%	3.46 / 2.22%	0.54 / 0.50%	6.20 / 3.27%
GalaxeaVLA (G0) (Jiang et al., 2025)	4.53 / 2.83%	8.10 / 3.83%	12.60 / 5.58%	3.17 / 1.89%	0.70 / 0.67%	5.82 / 2.96%
LingBot-VLA (Wu et al., 2026)	6.71 / 4.28%	5.33 / 1.83%	10.89 / 5.25%	3.82 / 2.78%	0.72 / 0.67%	5.50 / 2.96%
EventVLA (Yang et al., 2026a)	3.94 / 1.94%	10.13 / 5.75%	5.05 / 0.83%	4.92 / **4.78%**	0.80 / 0.75%	4.97 / 2.81%
AHA-WAM (Cai et al., 2026a)	5.79 / 3.28%	5.86 / 2.42%	8.61 / 2.67%	2.97 / 2.78%	0.88 / 0.83%	4.82 / 2.39%
ABot-M0 (Yang et al., 2026c)	5.73 / 3.50%	5.50 / 1.75%	3.96 / 0.50%	2.44 / 2.22%	0.72 / 0.67%	3.67 / 1.73%
Fast-WAM (Yuan et al., 2026b)	2.34 / 1.11%	1.96 / 0.00%	9.14 / 5.17%	3.55 / 3.44%	0.42 / 0.42%	3.48 / 2.03%
π_0 (Black et al., 2024)	3.94 / 2.56%	3.56 / 0.75%	6.19 / 2.00%	3.47 / 2.11%	0.25 / 0.25%	3.48 / 1.53%
GR00T-N1.7 (Bjorck et al., 2025)	2.16 / 1.22%	2.54 / 0.67%	8.30 / 3.58%	1.06 / 0.89%	0.18 / 0.17%	2.85 / 1.31%
InternVLA-A1 (Cai et al., 2026b)	2.87 / 1.83%	3.00 / 0.92%	4.79 / 1.17%	1.58 / 1.33%	0.17 / 0.17%	2.48 / 1.08%
SmolVLA (Single Task) (Shukor et al., 2025)	1.69 / 1.22%	2.87 / 0.33%	1.22 / 0.25%	3.35 / 2.44%	0.00 / 0.00%	1.83 / 0.85%
LDA-1B (Lyu et al., 2026)	0.71 / 0.17%	3.21 / 0.50%	1.92 / 0.08%	2.08 / 1.78%	0.00 / 0.00%	1.58 / 0.51%
MolmoAct2 (Fang et al., 2026)	0.39 / 0.06%	0.45 / 0.00%	2.32 / 0.00%	1.02 / 1.00%	0.91 / 0.83%	1.02 / 0.38%
GO-1 (Bu et al., 2025)	1.58 / 1.22%	1.45 / 0.42%	1.13 / 0.25%	0.70 / 0.67%	0.08 / 0.08%	0.99 / 0.53%
ACT (Single-Task) (Zhao et al., 2023)	0.69 / 0.56%	0.85 / 0.00%	1.73 / 0.92%	1.65 / 0.13%	0.00 / 0.00%	0.98 / 0.32%
H-RDT (Bi et al., 2026b)	0.49 / 0.22%	0.41 / 0.00%	2.23 / 0.17%	0.12 / 0.11%	0.08 / 0.08%	0.67 / 0.12%
RDT-1B (Liu et al., 2025a)	0.56 / 0.33%	0.38 / 0.00%	1.13 / 0.00%	0.49 / 0.33%	0.00 / 0.00%	0.51 / 0.13%
DM0 (Yu et al., 2026a)	0.49 / 0.06%	0.61 / 0.00%	0.97 / 0.08%	0.20 / 0.11%	0.00 / 0.00%	0.45 / 0.05%
Dexora-1B (Zhang et al., 2026e)	0.49 / 0.11%	0.49 / 0.00%	0.82 / 0.00%	0.12 / 0.00%	0.01 / 0.00%	0.38 / 0.02%
A1 (Zhang et al., 2026b)	0.16 / 0.00%	0.09 / 0.00%	1.07 / 0.00%	0.00 / 0.00%	0.08 / 0.08%	0.28 / 0.02%
Spirit v1.5 (Team et al.)	0.80 / 0.50%	0.03 / 0.00%	0.11 / 0.00%	0.22 / 0.22%	0.00 / 0.00%	0.23 / 0.14%
TinyVLA (Wen et al., 2025)	0.03 / 0.00%	0.05 / 0.00%	0.67 / 0.00%	0.11 / 0.11%	0.25 / 0.25%	0.22 / 0.07%
OpenVLA-OFT (Kim et al., 2025)	0.04 / 0.00%	0.20 / 0.00%	0.70 / 0.00%	0.00 / 0.00%	0.08 / 0.08%	0.21 / 0.02%
Human Expert (Teleop)	90.05 / 87.83%	68.06 / 64.00%	83.63 / 74.25%	75.25 / 74.33%	85.13 / 79.75%	80.42 / 76.03%

5.2. Real-World Benchmark Leaderboard

We evaluate 10 representative robot manipulation policies on the RoboDojo Real-World Benchmark Leaderboard. As a human-level reference, we additionally invite expert human teleoperators to perform real-world teleoperation using the same homogeneous leader-follower setup and task constraints as policy evaluation. Each teleoperator has more than 1,500 hours of real-world robot teleoperation experience and participated in collecting the RoboDojo Real-World Benchmark dataset.

For each evaluated policy, we train one random seed for each robot embodiment and evaluate it on the corresponding real-world tasks. The benchmark contains 18 tasks across three embodiments, with 10 trials per task, resulting in 180 real-world evaluation trials per policy. All evaluations are conducted through the RoboDojo-RealEval platform under the standardized scene reset procedure, evaluation protocol, and deployment interface described in Section 4.2. We report both task success rate and score, and compute aggregate performance across all real-world tasks.

The real-world leaderboard is frozen for this paper on *3 Jul. 2026*, while the latest results will be continuously updated at https://RoboDojo-Benchmark.com/Leaderboard. Detailed training configurations, hyperparameters, and policy-specific implementation details for all evaluated models are provided in Appendix K.

6. Experiments

Beyond the leaderboard results in Section 5, our experiments further analyze what RoboDojo reveals about current robot manipulation policies and evaluate the benchmark system itself. We focus on three questions: **(1)** what limitations RoboDojo reveals, by analyzing policy failures across capability dimensions and task types, and identifying key directions for future generalist manipulation policies; **(2)** how efficiently RoboDojo supports benchmark execution, by examining evaluation

Table 2: **RoboDojo Real-World Benchmark Leaderboard**. Each cell reports score / success rate. For each embodiment-specific task, embodiment average, and overall average, rankings among evaluated policies are marked as **Best** and **Second Best**; ties are marked consistently. Human teleoperation is reported as a reference and is excluded from policy ranking. (3 Jul. 2026, continuously updated at RoboDojo-Benchmark.com/Leaderboard.)

Policy	Embodiment	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Emb. Avg.	Overall Avg.
$\pi_{0.5}$ (Intelligence et al., 2025)	ARX X5	**24.6** / **20.0**	**1.8** / 0.0	**25.8** / **10.0**	**47.0** / **20.0**	**40.0** / **20.0**	**26.8** / **10.0**	**27.7** / **13.3**	**22.9** / **12.8**
	Piper	**10.0** / **10.0**	**28.0** / 0.0	**10.0** / **10.0**	0.0 / 0.0	**72.0** / **60.0**	**72.0** / **50.0**	**32.0** / **21.7**	
	Piper X	0.0 / 0.0	**14.8** / **10.0**	0.0 / 0.0	**29.5** / **10.0**	**7.5** / 0.0	3.0 / 0.0	**9.1** / **3.3**	
InternVLA-A1 (Cai et al., 2026b)	ARX X5	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	**48.0** / **20.0**	12.0 / 0.0	10.0 / 3.3	**12.0** / **7.2**
	Piper	0.0 / 0.0	7.3 / 0.0	0.0 / 0.0	0.0 / 0.0	**73.0** / **70.0**	**59.0** / **40.0**	**23.2** / **18.3**	
	Piper X	0.0 / 0.0	0.0 / 0.0	**4.0** / 0.0	2.0 / 0.0	0.0 / 0.0	**10.0** / 0.0	**2.7** / 0.0	
GalaxeaVLA (G0) (Jiang et al., 2025)	ARX X5	1.0 / 0.0	0.0 / 0.0	3.0 / 0.0	6.0 / 0.0	0.0 / 0.0	10.3 / 0.0	3.4 / 0.0	9.0 / 4.4
	Piper	0.0 / 0.0	**32.7** / **10.0**	**13.3** / **10.0**	0.0 / 0.0	56.0 / 50.0	30.0 / 10.0	22.0 / 13.3	
	Piper X	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	**4.0** / 0.0	0.0 / 0.0	6.0 / 0.0	1.7 / 0.0	
Xiaomi-Robotics-0 (Cai et al., 2026c)	ARX X5	**24.0** / **20.0**	0.0 / 0.0	3.0 / 0.0	4.0 / 0.0	**40.0** / **20.0**	**19.0** / **10.0**	**15.0** / **8.3**	7.9 / 3.9
	Piper	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	23.0 / 20.0	22.7 / 0.0	7.6 / 3.3	
	Piper X	0.0 / 0.0	0.7 / 0.0	**4.0** / 0.0	0.0 / 0.0	0.0 / 0.0	2.7 / 0.0	1.2 / 0.0	
X-VLA (Zheng et al., 2025)	ARX X5	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	2.0 / 0.0	20.7 / **10.0**	18.0 / 0.0	6.8 / 1.7	7.6 / 3.3
	Piper	0.0 / 0.0	5.3 / 0.0	0.0 / 0.0	0.0 / 0.0	53.0 / 50.0	37.0 / 0.0	15.9 / 8.3	
	Piper X	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.7 / 0.0	0.1 / 0.0	
GR00T-N1.7 (Bjorck et al., 2025)	ARX X5	10.0 / **10.0**	0.0 / 0.0	**9.0** / 0.0	14.7 / 0.0	16.0 / 0.0	6.0 / 0.0	9.3 / 1.7	5.9 / 1.7
	Piper	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	10.0 / 10.0	31.0 / 10.0	6.8 / 3.3	
	Piper X	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	**8.7** / 0.0	1.4 / 0.0	
π_0 (Black et al., 2024)	ARX X5	9.0 / 0.0	0.0 / 0.0	3.0 / 0.0	**18.0** / 0.0	0.0 / 0.0	18.7 / 0.0	8.1 / 0.0	5.8 / 1.7
	Piper	0.0 / 0.0	6.0 / 0.0	0.0 / 0.0	0.0 / 0.0	42.0 / 30.0	6.0 / 0.0	9.0 / 5.0	
	Piper X	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	2.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.3 / 0.0	
StarVLA-α (Ye et al., 2026b)	ARX X5	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	4.1 / 1.7
	Piper	0.0 / 0.0	14.0 / 0.0	0.0 / 0.0	0.0 / 0.0	39.0 / 30.0	18.7 / 0.0	12.0 / 5.0	
	Piper X	0.0 / 0.0	**2.7** / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.5 / 0.0	
Spirit v1.5 (Team et al.)	ARX X5	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	1.6 / 0.6
	Piper	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	10.0 / 10.0	14.0 / 0.0	4.0 / 1.7	
	Piper X	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	4.0 / 0.0	0.7 / 0.0	
DM0 (Yu et al., 2026a)	ARX X5	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0
	Piper	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	
	Piper X	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	
Human Expert (Teleop)	ARX X5	100.0 / 100.00	100.0 / 100.00	100.0 / 100.00	100.0 / 100.00	100.0 / 100.00	100.0 / 100.00	100.0 / 100.00	100.0 / 100.00
	Piper	100.0 / 100.00	100.0 / 100.00	100.0 / 100.00	100.0 / 100.00	100.0 / 100.00	100.0 / 100.00	100.0 / 100.00	
	Piper X	100.0 / 100.00	100.0 / 100.00	100.0 / 100.00	100.0 / 100.00	100.0 / 100.00	100.0 / 100.00	100.0 / 100.00	

Task order. **ARX X5**: `cover_blocks`, `make_bread`, `make_food`, `pack_and_pour_fruit`, `store_in_safe`, `insert_tubes`. **Piper**: `stack_and_cover_blocks`, `fill_pen_holder`, `put_objects_into_basket`, `insert_charger`, `stack_bowls`, `stand_up_bottles`. **Piper X**: `classify_objects`, `disassemble_LEGO`, `hang_mugs`, `pack_objects_into_backpack`, `sweep_blocks`, `cap_pen`.

efficiency in both heterogeneous parallel simulation and standardized real-world testing; and (3) how reproducible RoboDojo evaluation is, by studying simulation-side stability across repeated seeds and real-world consistency under standardized RoboDojo-RealEval conditions.

6.1. Analysis of RoboDojo Simulation Performance

Based on the simulation leaderboard in Section 5.1, we summarize key findings revealed by RoboDojo. Our analysis focuses on the current capabilities and limitations of generalist robot manipulation policies, including their performance gap from expert teleoperation and the remaining challenges toward robust general-purpose manipulation.

Finding 1: RoboDojo reveals a substantial gap toward balanced generalist robot manipulation. Table 1 summarizes the performance of current generalist robot manipulation policies on the RoboDojo simulation benchmark. Among the evaluated policies, Hy-Embodied-0.5-VLA, Spatial Forcing, $\pi_{0.5}$, X-VLA, Xiaomi-Robotics-0, X-WAM, GigaWorld-Policy, and StarVLA-α form the current leading group in terms of average performance. However, their results remain clustered in a low-score regime. Even the best-performing policy achieves only an 8.80% average success rate and a 13.07 average score, far below human experts, who reach a 76.03% success rate and an 80.42 score under the same evaluation protocol. This gap indicates that, although the tasks are feasible for human operators, current policies remain far from reliable task completion

Figure 8: **Domain randomization in RoboDojo.** We visualize the effects of domain randomization in simulation, including variations in background, lighting, clutter layout, object appearance, and scene configuration. These randomized environments increase visual diversity and reduce overfitting to a fixed simulation setting.

Table 3: **Policy performance under standard and randomized visual settings.** Each cell reports Standard Score / Random Score, with the relative score drop shown in parentheses. The drop is computed as $(\text{Standard} - \text{Random})/\text{Standard}$.

Policy-wise Score (Standard / Random, Relative Drop)							
Hy-Embodied-0.5-VLA	**Spatial Forcing**	$\pi_{0.5}$	**X_VLA**	**Xiaomi-Robotics-0**	**X_WAM**	**LingBot-VLA**	**AHA-WAM**
21.98/1.57 (92.9%)	21.25/6.98 (67.2%)	20.92/5.82 (72.2%)	17.92/3.04 (83.0%)	13.81/1.05 (92.4%)	11.24/3.54 (68.5%)	10.87/2.55 (76.5%)	10.32/1.26 (87.8%)
GigaWorldPolicy	**ABot-M0**	**GalaxeaVLA**	**starVLA**	π_0	**EventVLA**	**InternVLA-A1**	**FastWAM**
10.28/0.41 (96.0%)	9.19/2.26 (75.4%)	8.71/0.36 (95.9%)	7.53/0.33 (95.6%)	7.18/0.71 (90.1%)	6.66/1.22 (81.7%)	5.22/0.51 (90.2%)	4.33/0.34 (92.1%)
GR00T-N1.7	**SmolVLA**	**GO1**	**Spirit_v15**	**ACT**	**LDA-1B**	**Dexbotic-DM0**	**H-RDT**
3.97/0.35 (91.2%)	3.28/0.09 (97.3%)	3.16/0.01 (99.7%)	1.60/0.00 (100.0%)	1.37/0.00 (100.0%)	1.29/0.12 (90.7%)	0.93/0.04 (95.7%)	0.89/0.09 (89.9%)
RDT-1B	**Dexora-1B**	**MolmoACT2**	**A1**	**TinyVLA**	**OpenVLA-OFT**		
0.82/0.29 (64.6%)	0.80/0.18 (77.5%)	0.73/0.04 (94.5%)	0.28/0.04 (85.7%)	0.06/0.01 (83.3%)	0.03/0.06 (−100.0%)		

across diverse manipulation scenarios.

Beyond this overall gap, RoboDojo further reveals that current policies lack balanced capability development across dimensions. Different methods exhibit relative strengths in different areas: Spatial Forcing performs strongly on Generalization, X-VLA leads on Precision, $\pi_{0.5}$ is competitive on Open tasks, while Hy-Embodied-0.5-VLA achieves the best Long-Horizon, Memory, and overall performance. Nevertheless, these strengths remain dimension-specific and do not consistently translate across the full benchmark. Current policies perform relatively better on Generalization and Long-Horizon tasks, suggesting partial progress in visual grounding, goal localization, and multi-step execution. However, this advantage is only relative: even on these stronger dimensions, the best success rates remain below 15%. Precision, Memory, and Open tasks continue to expose more severe bottlenecks in fine-grained control, persistent scene understanding, and open-ended task execution. These results show that RoboDojo provides diagnostic signals beyond a single aggregate success rate, revealing fragmented progress across individual capabilities and a substantial gap toward balanced generalist robot manipulation.

Finding 2: Scene-level randomization causes broad performance collapse, while visual-spatial grounding only partially improves robustness.

The Generalization dimension in RoboDojo evaluates policy robustness under visual and scene-level distribution shifts, including changes in object instances, layouts, clutter, lighting, and visual backgrounds. As shown in Table 1, Spatial Forcing achieves the strongest overall performance on this dimension, with a 9.33% success rate and a 14.12 score. The Standard–Random comparison in Table 3 further shows that the difficulty does not come from a few isolated hard cases; instead, scene randomization causes a broad performance collapse across nearly all policies.

A key observation is that strong performance in the Standard setting does not necessarily transfer to randomized scenes. Among the higher-performing methods, Hy-Embodied-0.5-VLA obtains the highest Standard score of 21.98, but drops to 1.57 under Random, corresponding to a 92.9% relative drop. In contrast, Spatial Forcing improves over its $\pi_{0.5}$ base model in both absolute Random score (6.98 vs. 5.82) and relative score retention, reducing the drop from 72.2% to 67.2%. This suggests that explicit 3D spatial grounding and spatial representation alignment can improve robustness to scene-level variations, especially when object poses, spatial layouts, and visual backgrounds change.

However, this improvement remains limited in absolute terms. Even the strongest Random score is only 6.98, and most competitive policies lose the majority of their Standard-setting performance after randomization. This indicates that current generalist manipulation policies remain highly sensitive to visual and spatial distribution shifts. Importantly, the failure is not merely a high-level recognition problem: even when policies can identify task-relevant objects or goals, distribution shifts may still degrade metric localization, action grounding, trajectory generation, and recovery from off-nominal states. Therefore, visual-spatial grounding is helpful for scene-level generalization, but reliable manipulation under randomized scenes also requires stable low-level control, robust closed-loop correction, and recovery-oriented policy behavior.

Finding 3: Current policies make partial progress on structured long-horizon tasks, but reliable skill composition

remains difficult. Long-Horizon is one of the relatively stronger dimensions in RoboDojo for leading policies. Hy-Embodied-0.5-VLA achieves the best performance, with a 14.92% success rate and a 25.74 score, followed closely by $\pi_{0.5}$ with a 14.67% success rate and a 23.54 score, and Spatial Forcing with a 14.58% success rate and a 23.26 score. Compared with Precision, Memory, and Open tasks, these results suggest that current policies can make more progress on structured multi-step workflows with clear intermediate goals and visually grounded substeps. Nevertheless, even the best policy succeeds in fewer than 15% of Long-Horizon episodes, indicating that long-horizon manipulation remains far from reliable.

The gap between scores and success rates suggests that many policies can complete partial stages but fail to consistently reach the final goal. This reflects a central challenge in long-horizon manipulation: individual skills do not automatically compose into robust multi-step behavior. Policies must maintain task progress, select appropriate subtasks, switch stages at the right time, and recover from intermediate errors. Small execution errors can accumulate over extended sequences, causing later stages to fail even when early steps are partially successful.

The leading results further indicate that stronger action priors, embodied pretraining, and spatial grounding can help long-horizon execution. Hy-Embodied-0.5-VLA may benefit from large-scale embodied pretraining and temporally structured action prediction, while Spatial Forcing and $\pi_{0.5}$ suggest the value of spatial grounding and strong base policy training. However, these advantages remain limited. Overall, current policies show partial progress on structured long-horizon execution, but still lack reliable skill composition, stage-level decision making, and error recovery.

Finding 4: Precision remains a major bottleneck due to limited contact-aware control and weak off-trajectory correction. The Precision dimension exposes a distinct failure mode from Generalization and Long-Horizon tasks. Policies must not only recognize task-relevant objects and target poses, but also generate spatially accurate, temporally smooth, and contact-sensitive actions under tight tolerances. As shown in Table 1, X-VLA achieves the best Precision performance, with a 12.00% success rate and an 18.32 score, followed by Spatial Forcing with a 10.58% success rate and a 17.33 score, and Hy-Embodied-0.5-VLA with an 8.00% success rate and a 13.81 score. However, even the strongest models remain far from reliable precision manipulation.

The Precision ranking differs from the overall leaderboard: Hy-Embodied-0.5-VLA obtains the best average performance, but is surpassed by X-VLA and Spatial Forcing on precision tasks. This suggests that stronger global task execution does not automatically yield accurate local control. Precision tasks require metric spatial reasoning, fine-grained motion generation, and stable contact transitions. Although stronger spatial grounding or action alignment may provide relative advantages, current policies still lack sufficiently robust 3D localization, smooth action prediction, and closed-loop contact control.

Our rollouts further suggest that many precision failures arise from weak state-conditioned correction rather than task misunderstanding. For example, in `insert_key`, after failing to hand over or align the key, many policies still continue executing the subsequent insertion motion, indicating that they often follow an open-loop action sequence without verifying whether the current state satisfies the precondition for the next stage. We also observe action jitter in several policies across both end-effector and joint-space control interfaces, suggesting that such instability is not specific to a particular action representation, but is more likely related to insufficient temporal smoothness in action prediction. These results indicate that reliable precision manipulation requires smoother low-level action priors, contact-aware feedback, and correction-oriented training signals for recovering from off-trajectory states.

Finding 5: Memory mechanisms help, but memory-conditioned execution remains the bottleneck. The Memory dimension shows that history- or memory-conditioned designs can improve performance, but are still far from reliable. Hy-Embodied-0.5-VLA achieves the best result, with a 12.11% success rate and a 13.37 score, while EventVLA also benefits from its KEM-based memory design, reaching a 4.78% success rate and a 4.92 score. However, the gap between these models suggests that explicit memory structure alone is insufficient. EventVLA stores sparse long-horizon evidence more explicitly, but our rollouts indicate that many failures arise from downstream manipulation errors or incorrect subtask execution rather than the complete absence of memory cues. In contrast, Hy-Embodied-0.5-VLA may benefit from stronger embodied pretraining and action priors, which help convert historical context into executable behavior. These results suggest that the key challenge is not only remembering past observations, but also using memory to guide state-conditioned decisions and robust action execution.

World-model-style prediction provides another form of implicit temporal memory. X-WAM achieves a 4.67% success rate and a 6.32 score on Memory tasks, slightly below EventVLA in success rate but higher in score. This suggests that predictive pretraining can help encode temporal continuity, object permanence, and task progress, leading to stronger partial completion. However, predictive models do not necessarily preserve or retrieve sparse historical evidence after it disappears from the current observation. Their limited final success rates indicate that temporal prediction alone is insufficient for tasks requiring reliable sparse-evidence recall and memory-conditioned control. Overall, RoboDojo suggests that future policies should

combine predictive temporal modeling, explicit memory supervision, and action learning that directly conditions execution on remembered evidence.

Finding 6: Open-semantic manipulation remains largely unsolved. The Open dimension is the most challenging split in RoboDojo. Current policies achieve near-zero performance on this dimension, and even the best-performing model, $\pi_{0.5}$, reaches only a 1.67% success rate and a 1.98 score. This result exposes a critical limitation of current generalist manipulation policies: they remain poorly equipped for tasks where language instructions require semantic interpretation beyond closed-set demonstrations. In real-world settings, users may specify novel goals, refer to objects by function or intent, or expect robots to recombine learned skills in unfamiliar ways. Unlike closed-set imitation tasks, Open tasks cannot be solved by simply matching familiar instructions to memorized action patterns.

These results suggest that current policies still lack robust semantic-to-action grounding. Although strong vision-language backbones can improve high-level perception and language understanding, they often fail to align open-ended instructions, visual affordances, and executable manipulation actions. In particular, policies must infer task intent, identify relevant objects or functional relations, select appropriate skills, and execute them under physical constraints. The near-zero Open performance indicates that this full semantic-to-action pipeline remains highly fragile. Future generalist manipulation policies therefore require stronger multimodal alignment, open-vocabulary affordance grounding, compositional skill learning, and execution-aware training signals that connect semantic goals to reliable robot behaviors.

6.2. Analysis of RoboDojo Real-World Performance

Before presenting the findings, we analyze the real-world results from three aspects: the gap to human teleoperation, the mismatch between simulation and real-world rankings, and deployment-specific failures such as action jitter and unsafe behaviors. These analyses show that RoboDojo-RealEval complements simulation by exposing physical-world bottlenecks that are difficult to capture in simulation alone.

Finding 1: Real-world deployment remains unreliable, with partial progress rarely translating into task completion. Table 2 shows that RoboDojo-RealEval remains highly challenging for current generalist manipulation policies. The best-performing policy, $\pi_{0.5}$, achieves only a 12.8% overall success rate and a 22.9 score across 18 real-world tasks, while human teleoperation reaches a 100.0% success rate and a 100.0 score on all tasks and embodiments. This gap shows that the tasks are feasible under the standardized protocol, yet current policies remain far from reliable physical deployment.

The higher scores relative to success rates further indicate that current policies can often make partial progress but fail to complete the full task. For example, $\pi_{0.5}$, InternVLA-A1, and GalaxeaVLA obtain scores of 22.9, 12.0, and 9.0, respectively, but their success rates remain only 12.8%, 7.2%, and 4.4%. This suggests that real-world failures are driven not only by task misunderstanding, but also by execution bottlenecks such as precise alignment, contact handling, sub-step transition, and recovery from intermediate errors. Reliable real-world manipulation therefore requires closed-loop execution that can convert partial progress into final task success under physical uncertainty.

Finding 2: Simulation performance and real-world deployability are only partially aligned. Since RoboDojo-RealEval currently evaluates a subset of the simulation leaderboard, we focus on policies evaluated in both settings. Among these overlapping policies, the two leaderboards show partial but not complete alignment. $\pi_{0.5}$ remains the strongest policy in RoboDojo-RealEval and is also among the leading methods in simulation, suggesting that simulation performance captures important aspects of general manipulation capability. However, the relative ordering changes for several policies: InternVLA-A1 and GalaxeaVLA achieve stronger real-world positions than their simulation rankings would suggest, while some policies with competitive simulation performance do not preserve the same advantage after physical deployment.

This mismatch should be interpreted carefully. RoboDojo is not designed as a paired sim-to-real transfer benchmark with one-to-one matched simulation and real-world task distributions. Instead, the two settings stress complementary aspects of generalist manipulation policies. Simulation provides scalable capability diagnosis across generalization, precision, long-horizon execution, memory, and open-semantic grounding, while RoboDojo-RealEval measures whether policies remain executable and reliable under physical deployment constraints.

Real-world evaluation introduces factors that are difficult to fully capture in simulation, including perception noise, camera and robot calibration errors, actuation latency, controller-specific dynamics, contact instability, and accumulated execution drift. These factors can cause policies with strong simulation performance to fail if their actions are jittery, poorly calibrated, insufficiently contact-aware, or unable to recover from small deviations. The partial alignment therefore supports the sim-and-real design of RoboDojo: simulation enables efficient large-scale capability analysis, while real-world evaluation exposes deployment-specific bottlenecks that cannot be inferred from simulation alone.

Finding 3: Real-world evaluation exposes execution instability and safety-critical behaviors beyond aggregate success metrics. Real-world evaluation reveals deployment-specific failure modes that are not fully captured by aggregate success rates or simulation scores. Across several policies, we observe action jitter, oscillatory motions, repeated ineffective actions, contact instability, and occasional safety-critical behaviors. These failures indicate that physical deployment depends not only on high-level task understanding, but also on the temporal stability and safety of low-level action execution.

For example, DM0 frequently produces unstable control signals during deployment, leading to erratic motions that require close monitoring or safety intervention. This distinction is important: in simulation, unstable actions may only reduce task scores, whereas on physical robots they can introduce hardware risk, unsafe contacts, or unintended interactions with the environment. Therefore, final task success alone is insufficient to characterize real-world deployment quality.

These observations suggest that future generalist manipulation policies should optimize deployment-oriented properties beyond task completion, including action smoothness, contact-aware feedback, bounded control behavior, and recovery from off-trajectory states. RoboDojo-RealEval therefore provides complementary diagnostic value by revealing whether a policy is not only capable in principle, but also stable and safe enough for physical execution.

6.3. Evaluation Efficiency

In this section, we evaluate the efficiency of RoboDojo in both simulation and real-world settings. For simulation, we measure the throughput improvement brought by heterogeneous parallel simulation under zero-action rollouts and large-policy inference. For real-world evaluation, we measure the wall-clock time required to complete physical trials, including scene reset, policy inference, and robot execution. These results quantify the practical cost of running RoboDojo at benchmark scale.

6.3.1. Simulation Evaluation Efficiency

To evaluate the efficiency gain from heterogeneous parallel simulation, we conduct controlled experiments under five representative settings: RoboDojo with heterogeneous or non-heterogeneous parallel simulation under zero-action rollouts, RoboDojo with heterogeneous or non-heterogeneous parallel simulation coupled with $\pi_{0.5}$ policy inference, and RoboTwin 2.0 with zero-action rollouts. All simulation and rendering workloads are executed on 8×RTX 4090 GPUs. In our setup, each RTX 4090 runs one RoboDojo simulation process with 10 parallel environments, while RoboTwin 2.0 runs two simulation processes per GPU. Notably, RoboTwin 2.0 renders three camera views at a resolution of 320×240, whereas RoboDojo renders at 640×480, resulting in a higher per-frame rendering workload. For non-zero-action RoboDojo settings, $\pi_{0.5}$ inference is performed on an A800 server within the same local network. The zero-action setting measures raw simulation throughput without policy inference overhead, whereas the $\pi_{0.5}$ setting evaluates end-to-end efficiency under a realistic remote policy-deployment scenario.

As shown in Table 4, heterogeneous parallel simulation substantially improves evaluation throughput. Under zero-action rollouts, RoboDojo achieves 77.4 interactions/s, compared with 40.0 interactions/s under non-heterogeneous parallel simulation, yielding a $1.94\times$ speedup. When $\pi_{0.5}$ inference is included, heterogeneous parallel simulation achieves 64.0 interactions/s, outperforming the non-heterogeneous setting by $1.63\times$. These results show that the proposed infrastructure improves both raw simulation throughput and practical end-to-end evaluation efficiency. Compared with RoboTwin 2.0 zero-action rollouts, RoboDojo achieves higher normalized throughput while supporting more diverse tasks and scenes. This efficiency enables large-scale policy evaluation and rapid feedback across comprehensive capability dimensions.

Table 4: **Simulation evaluation efficiency.** All simulation and rendering workloads are executed on 8×RTX 4090 GPUs. RoboDojo runs one simulation process with 10 parallel environments per RTX 4090, while RoboTwin 2.0 runs two simulation processes per GPU. For non-zero-action RoboDojo settings, $\pi_{0.5}$ policy inference is performed on an A800 server within the same local network.

Benchmark	Evaluation Setting	Frames	Time	Avg. Speed
RoboDojo	Heterogeneous parallel simulation, zero action	1,640,000	5h 53m	77.4 interactions/s
RoboDojo	Non-heterogeneous parallel simulation, zero action	1,640,000	11h 22m	40.0 interactions/s
RoboDojo	Heterogeneous parallel simulation + $\pi_{0.5}$ inference	1,640,000	7h 07m	64.0 interactions/s
RoboDojo	Non-heterogeneous parallel simulation + $\pi_{0.5}$ inference	1,640,000	11h 38m	39.2 interactions/s
RoboTwin 2.0	Zero-action rollout	3,100,000	19h 19m	44.6 interactions/s

6.3.2. Real-World Evaluation Efficiency

We further evaluate the efficiency of RoboDojo-RealEval using $\pi_{0.5}$ as the test policy. The policy server is deployed on an RTX 4090 server within the same local-area network as the real-world robot platform. For each task, we measure the wall-clock time required to complete 10 full evaluation trials, including scene reset, policy inference, and robot execution. All trials are executed to completion without manual interruption or early stopping, making the reported time a conservative estimate of the real-world evaluation cost.

As shown in Table 5, RoboDojo-RealEval completes the 18-task real-world evaluation in 202.0 minutes, corresponding to approximately 3.4 hours for 180 physical trials. On average, each task requires 11.2 minutes for 10 trials. These results indicate that the standardized scene reset procedure, local RTX 4090 policy deployment, and unified evaluation interface substantially reduce the operational overhead of real-world benchmarking, enabling RoboDojo-RealEval to provide relatively rapid feedback while preserving full physical evaluation.

Table 5: **Real-world evaluation time for each task.** Each entry reports the wall-clock time required to complete 10 full evaluation trials, including scene reset, RTX 4090 policy inference within the same local-area network, and robot execution.

Task	Time (min)	Task	Time (min)	Task	Time (min)
connect_charger	13.5	stand_up_bottles	7.2	cap_pen	6.1
stack_bowls	7.3	put_objects_into_basket	20.5	pack_and_pour_fruit	11.1
stack_and_cover_blocks	9.9	hang_mugs	9.2	store_in_safe	9.8
fill_pen_holder	10.7	pack_objects_into_backpack	12.1	make_food	24.0
sweep_blocks	6.0	classify_objects	15.5	insert_tubes	9.0
disassemble_LEGO	5.7	cover_blocks	10.8	make_bread	13.6
				Total	202.0

6.4. Evaluation Stability

We evaluate the stability of RoboDojo by measuring performance variation across both simulation and real-world evaluation settings. In simulation, we test whether the same policy obtains consistent results across different simulation workers and GPU deployments. In the real world, we examine cross-system consistency across RoboDojo-RealEval platforms, where hardware setup, lighting conditions, camera placement, and scene layout replay are standardized. This study assesses whether RoboDojo can provide stable and reliable scores for leaderboard comparison and remote real-world evaluation.

6.4.1. Simulation Evaluation Stability

Although RoboDojo uses standardized task configurations and evaluation protocols, Isaac Sim rendering and low-level simulation execution may still introduce minor nondeterminism across GPU devices. To evaluate the cross-GPU consistency of the simulation benchmark, we conduct stability experiments on three RTX 4090 GPUs. Specifically, we use layout 0 of each task and evaluate three representative policies, including $\pi_{0.5}$, Xiaomi-Robotics-0, and GalaxeaVLA, with three random seeds on each GPU.

Table 7 reports the cross-GPU standard deviation of success rate and score. Overall, RoboDojo produces consistent simulation evaluation results across different GPUs. Across all policies and capability dimensions, the largest success-rate standard deviation is only 1.1 percentage points, and the largest score standard deviation is 1.07. For the overall average, the standard deviation is at most 0.5 percentage points in success rate and 0.49 in score. These results indicate that, under standardized evaluation layouts, RoboDojo yields stable simulation scores across GPU devices, making it suitable for fair leaderboard comparison despite minor nondeterminism from simulator rendering and low-level execution.

Table 6: **Real-world evaluation stability.** We report the standard deviation across three repeated runs. Full per-task results are in Appendix Table 9.

Policy	Overall		Max Task	
	SR	Score	SR	Score
$\pi_{0.5}$	1.0	0.5	23.1	12.9
GalaxeaVLA	1.2	1.0	15.3	16.6
InternVLA-A1	1.3	1.2	17.3	10.4

6.4.2. Real-World Evaluation Stability

To examine the repeatability of RoboDojo-RealEval, we repeat real-world evaluation for three representative leaderboard policies: $\pi_{0.5}$, InternVLA-A1, and GalaxeaVLA. Each policy is evaluated for three independent rounds under the same RoboDojo-RealEval protocol, covering all 18 real-world tasks across three robot embodiments.

Table 7: **Simulation evaluation stability across GPUs.** We evaluate $\pi_{0.5}$, Xiaomi-Robotics-0, and GalaxeaVLA on three RTX 4090 GPUs using layout 0 of each task, with three random seeds on each GPU. Each cell reports the cross-GPU standard deviation in the format of success rate / score. Success-rate standard deviations are reported in percentage points.

Policy	Generalization	Precision	Long-Horizon	Memory	Open	Average
$\pi_{0.5}$	0.4 / 0.46	0.3 / 0.15	0.8 / 0.54	0.2 / 0.18	0.3 / 0.31	0.1 / 0.19
Xiaomi-Robotics-0	0.3 / 0.31	1.1 / 1.07	0.7 / 0.82	0.5 / 0.72	0.2 / 0.12	0.5 / 0.49
GalaxeaVLA	0.4 / 0.45	0.5 / 1.06	0.6 / 0.68	0.7 / 0.54	0.2 / 0.13	0.2 / 0.37

Table 6 summarizes the aggregate repeated-evaluation stability, and the full per-task breakdown is provided in Appendix Table 9. Overall, RoboDojo-RealEval produces stable aggregate evaluation results. Across the three policies, the standard deviation of the overall success rate is at most 1.3 percentage points, and the standard deviation of the overall score is at most 1.2 points. This indicates that the overall leaderboard results are not overly sensitive to a single evaluation round.

At the task level, several tasks exhibit larger variance, especially when the number of physical trials is limited and the task involves contact-rich or multi-stage execution. Such variance is expected in real-world robot evaluation, where small differences in object placement, contact state, perception noise, and accumulated execution error can affect the final outcome. Nevertheless, these variations are largely averaged out at the aggregate level, supporting the repeatability of RoboDojo-RealEval under its standardized hardware setup, scene reset procedure, evaluation protocol, and scoring process.

7. Future Extensions

Figure 9: **Future extensions of RoboDojo.** RoboDojo will be continuously expanded to broader manipulation scenarios and embodiments, including dexterous hand manipulation, humanoid whole-body manipulation, tactile manipulation, and mobile manipulation.

RoboDojo is designed as an extensible benchmarking platform rather than a fixed task collection. In future releases, we will continuously expand RoboDojo toward broader scenarios, embodiments, and evaluation settings. As shown in Figure 9, we plan to introduce larger-scale benchmarks for dexterous hand manipulation, humanoid whole-body manipulation, tactile manipulation, and mobile manipulation. For each direction, RoboDojo will support multiple robot embodiments to improve accessibility and enable fair comparison across different hardware platforms. By progressively extending the task suite, embodiment coverage, and sensing modalities, RoboDojo aims to evolve into a general-purpose evaluation platform for embodied manipulation.

8. Conclusion

We presented **RoboDojo**, a unified sim-and-real benchmark for evaluating generalist robot manipulation policies. RoboDojo includes 42 simulation tasks and 18 real-world tasks, covering diverse, long-horizon, precision-demanding, memory-dependent, and open-ended manipulation scenarios. The simulation benchmark supports efficient large-scale evaluation through heterogeneous parallel execution, enabling rapid feedback and fine-grained capability diagnosis. The real-world benchmark, built on RoboDojo-RealEval, provides standardized hardware configurations, scene reset procedures, evaluation protocols, and remote cloud-based access for reproducible physical assessment. Together with **XPolicyLab**, a unified standard and infrastructure for policy development and deployment, RoboDojo enables policies to be integrated, evaluated, and compared under a shared protocol.

Our experiments show that current robot policies remain far from reliable general-purpose manipulation. In simulation,

RoboDojo reveals insufficient capability coverage across key dimensions, including generalization, long-horizon execution, precise manipulation, memory, and open-ended task understanding, indicating that current models are not yet comprehensive across diverse manipulation requirements. In the real world, the evaluated policies still perform poorly on complex manipulation tasks, showing that current models are not yet robust enough to handle challenging physical deployment conditions. RoboDojo will continue to update its leaderboard with new policies, tasks, evaluation results, and community submissions, providing a challenging testbed, reproducible evaluation platform, and open benchmark for tracking progress toward more capable omni-manipulation policies.

References

[1] Pranav Atreya, Karl Pertsch, Tony Lee, Moo Jin Kim, Arhan Jain, Artur Kuramshin, Clemens Eppner, Cyrus Neary, Edward Hu, Fabio Ramos, et al. Roboarena: Distributed real-world evaluation of generalist robot policies. *arXiv preprint arXiv:2506.18123*, 2025.

[2] Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 35101–35113, 2026a.

[3] Hongzhe Bi, Lingxuan Wu, Tianwei Lin, Hengkai Tan, Zhizhong Su, Hang Su, and Jun Zhu. H-rdt: Human manipulation enhanced bimanual robotic manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 18135–18143, 2026b.

[4] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[5] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. pi_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[6] Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.

[7] Jisong Cai, Long Ling, Shiwei Chu, Zhongshan Liu, Jiayue Kang, Zhixuan Liang, Wenjie Xu, Yinan Mao, Weinan Zhang, Xiaokang Yang, et al. Aha-wam: Asynchronous horizon-adaptive world-action modeling with observation-guided context routing. *arXiv preprint arXiv:2606.09811*, 2026a.

[8] Junhao Cai, Zetao Cai, Jiafei Cao, Yilun Chen, Zeyu He, Lei Jiang, Hang Li, Hengjie Li, Yang Li, Yufei Liu, et al. Internvla-a1: Unifying understanding, generation and action for robotic manipulation. *arXiv preprint arXiv:2601.02456*, 2026b.

[9] Rui Cai, Jun Guo, Xinze He, Piaopiao Jin, Jie Li, Bingxuan Lin, Futeng Liu, Wei Liu, Fei Ma, Kun Ma, et al. Xiaomi-robotics-0: An open-sourced vision-language-action model with real-time execution. *arXiv preprint arXiv:2602.12684*, 2026c.

[10] Stéphane Caron, Yann De Mont-Marin, Rohan Budhiraja, Seung Hyeon Bang, Ivan Domrachev, Simeon Nedelchev, Peter Du, Adrien Escande, Joris Vaillant, Bruce Wingo, Santosh Patapati, Daniel San José Pro, and Nicolas Guillermo Marticorena Vidal. Pink: Python inverse kinematics based on Pinocchio, 2026. URL https://github.com/stephane-caron/pink.

[11] Baijun Chen, Weijie Wan, Tianxing Chen, Xianda Guo, Congsheng Xu, Yuanyang Qi, Haojie Zhang, Longyan Wu, Tianling Xu, Zixuan Li, et al. Univtac: A unified simulation platform for visuo-tactile manipulation data generation, learning, and benchmarking. *arXiv preprint arXiv:2602.10093*, 2026a.

[12] Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025a.

[13] Tianxing Chen, Yao Mu, Zhixuan Liang, Zanxin Chen, Shijia Peng, Qiangyu Chen, Mingkun Xu, Ruizhen Hu, Hongyuan Zhang, Xuelong Li, et al. G3flow: Generative 3d semantic flow for pose-aware and generalizable object manipulation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 1735–1744, 2025b.

[14] Tianxing Chen, Kaixuan Wang, Zhaohui Yang, Yuhao Zhang, Zanxin Chen, Baijun Chen, Wanxi Dong, Ziyuan Liu, Dong Chen, Tianshuo Yang, et al. Benchmarking generalizable bimanual manipulation: Robotwin dual-arm collaboration challenge at cvpr 2025 meis workshop. *arXiv preprint arXiv:2506.23351*, 2025c.

[15] Tianxing Chen, Yuran Wang, Mingleyang Li, Yan Qin, Hao Shi, Zixuan Li, Yifan Hu, Yingsheng Zhang, Kaixuan Wang, Yue Chen, et al. Rmbench: Memory-dependent robotic manipulation benchmark with insights into policy design. *arXiv preprint arXiv:2603.01229*, 2026b.

[16] Yiting Chen, Kenneth Kimble, Edward H Adelson, Tamim Asfour, Podshara Chanrungmaneekul, Sachin Chitta, Yash Chitambar, Ziyang Chen, Ken Goldberg, Danica Kragic, et al. Manipulationnet: An infrastructure for benchmarking real-world robot manipulation with physical skill challenges and embodied multimodal reasoning. *arXiv preprint arXiv:2603.04363*, 2026c.

[17] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion, March 2024.

[18] Haoquan Fang, Jiafei Duan, Donovan Clay, Sam Wang, Shuo Liu, Weikai Huang, Xiang Fan, Wei-Chuan Tsai, Shirui Chen, Yi Ru Wang, et al. Molmoact2: Action reasoning models for real-world deployment. *arXiv preprint arXiv:2605.02881*, 2026.

[19] Galaxea Team. Galaxea g0.5 technical report. 2026. URL https://opengalaxea.github.io/G05/.

[20] Ning Gao, Jinliang Zheng, Xing Gao, Haoxiang Ma, Hanqing Wang, Yukai Wang, Jiantong Chen, Zanxin Chen, Shujie Zhang, Mingda Jia, et al. Ebench: Elemental diagnosis of generalist mobile manipulation policies. *arXiv preprint arXiv:2606.18239*, 2026.

[21] Jun Guo, Qiwei Li, Peiyan Li, Zilong Chen, Nan Sun, Yifei Su, Heyun Wang, Yuan Zhang, Xinghang Li, and Huaping Liu. Unified 4d world action modeling from video priors with asynchronous denoising, 2026. URL https://arxiv.org/abs/2604.26694.

[22] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al.
$pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[23] Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, Vedant Choudhary, Foster Collins, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Maitrayee Dhaka, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachlan Groom, Haroun Habeeb, Hunter Hancock, Karol Hausman, Gashon Hussein, Victor Hwang, Brian Ichter, Connor Jacobsen, Szymon Jakubczak, Rowan Jen, Tim Jones, Gregg Kammerer, Ben Katz, Liyiming Ke, Mairbek Khadikov, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Brendon LeCount, Sergey Levine, Xinyu Li, Adrian Li-Bell, Vladislav Lialin, Zhonglin Liang, Wallace Lim, Yao Lu, Enyu Luo, Vishnu Mano, Nandan Marwaha, Aikys Mongush, Liam Murphy, Suraj Nair, Tyler Patterson, Karl Pertsch, Allen Z. Ren, Gavin Schelske, Charvi Sharma, Baifeng Shi, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Jiaming Tang, Jimmy Tanner, Shalom Tekeste, Marcel Torne, Kyle Vedder, Quan Vuong, Anna Walling, Haohuan Wang, Jason Wang, XuDong Wang, Chris Whalen, Samuel Whitmore, Blake Williams, Charles Xu, Sukwon Yoo, Lili Yu, Wuming Zhang, Zhuoyang Zhang, and Ury Zhilinsky. $\pi_{0.7}$: a steerable generalist robotic foundation model with emergent capabilities, 2026a. URL https://arxiv.org/abs/2604.15483.

[24] Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, et al. Pi 0.7: a steerable generalist robotic foundation model with emergent capabilities. *arXiv preprint arXiv:2604.15483*, 2026b.

[25] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. Rlbench: The robot learning benchmark & learning environment, 2019. URL https://arxiv.org/abs/1909.12271.

[26] Tao Jiang, Tianyuan Yuan, Yicheng Liu, Chenhao Lu, Jianning Cui, Xiao Liu, Shuiqi Cheng, Jiyang Gao, Huazhe Xu, and Hang Zhao. Galaxea open-world dataset and g0 dual-system vla model. *arXiv preprint arXiv:2509.00576*, 2025.

[27] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[28] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

[29] Zhiqian Lan, Yuxuan Jiang, Ruiqi Wang, Xuanbing Xie, Rongkui Zhang, Yicheng Zhu, Peihang Li, Tianshuo Yang, Tianxing Chen, Haoyu Gao, et al. Autobio: A simulation and benchmark for robotic automation in digital biology laboratory. *arXiv preprint arXiv:2505.14030*, 2025.

[30] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, et al. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning*, pages 80–93. PMLR, 2023.

[31] Fuhao Li, Wenxuan Song, Han Zhao, Jingbo Wang, Pengxiang Ding, Donglin Wang, Long Zeng, and Haoang Li. Spatial forcing: Implicit spatial representation alignment for vision-language-action model. *arXiv preprint arXiv:2510.12276*, 2025a.

[32] Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, et al. Simplevla-rl: Scaling vla training via reinforcement learning. *arXiv preprint arXiv:2509.09674*, 2025b.

[33] Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal world modeling for robot control, 2026a. URL https://arxiv.org/abs/2601.21998.

[34] Shalfun Li, Victor Yao, Charles Yang, Truth Qu, Regis Cheng, Ryan Yu, Howard Lu, Newton Von, Vincent Chen, Yohann Tang, et al. Wall-wm: Carving world action modeling at the event joints. *arXiv preprint arXiv:2606.01955*, 2026b.

[35] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation, 2024a. URL https://arxiv.org/abs/2405.05941.

[36] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, et al. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024b.

[37] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

[38] Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. In *International Conference on Learning Representations*, volume 2025, pages 29982–30009, 2025a.

[39] Yushan Liu, Shilong Mu, Xintao Chao, Zizhen Li, Yao Mu, Tianxing Chen, Shoujie Li, Chuqiao Lyu, Xiao-ping Zhang, and Wenbo Ding. Avr: Active vision-driven robotic precision manipulation with viewpoint and focal length optimization. *arXiv e-prints*, pages arXiv–2503, 2025b.

[40] Haoran Lu, Ruihai Wu, Yitong Li, Sijie Li, Ziyu Zhu, Chuanruo Ning, Yan Shen, Longzan Luo, Yuanpei Chen, and Hao Dong. Garmentlab: A unified simulation and benchmark for garment manipulation. In *Advances in Neural Information Processing Systems*, 2024.

[41] Haoran Lu, Songling Liu, Yue Chen, Guo Ye, Mutian Shen, Shuyang Yu, Yu Xiao, Jihai Zhao, Shang Wu, Jianshu Zhang, Xiangtian Gui, Chuye Hong, Yuran Wang, Maojiang Su, Jiayi Wang, Ruihai Wu, Zhaoran Wang, and Han Liu. Magicsim: A unified infrastructure for executable embodied interaction, 2026. URL https://arxiv.org/abs/2606.17511.

[42] Jiangran Lyu, Kai Liu, Xuheng Zhang, Haoran Liao, Yusen Feng, Wenxuan Zhu, Tingrui Shen, Jiayi Chen, Jiazhao Zhang, Yifei Dong, et al. Lda-1b: Scaling latent dynamics action model via universal embodied data ingestion. *arXiv preprint arXiv:2602.12215*, 2026.

[43] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.

[44] Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Muñoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, Lukasz Wawrzyniak, Milad Rakhsha, Alain Denzler, Eric Heiden, Ales Borovicka, Ossama Ahmed, Iretiayo Akinola, Abrar Anwar, Mark T. Carlson, Ji Yuan Feng, Animesh Garg, Renato Gasoto, Lionel Gulich, Yijie Guo, M. Gussert, Alex Hansen, Mihir Kulkarni, Chenran Li, Wei Liu, Viktor Makoviychuk, Grzegorz Malczyk, Hammad Mazhar, Masoud Moghani, Adithyavairavan Murali, Michael Noseworthy, Alexander Poddubny, Nathan

Ratliff, Welf Rehberg, Clemens Schwarke, Ritvik Singh, James Latham Smith, Bingjie Tang, Ruchik Thaker, Matthew Trepte, Karl Van Wyk, Fangzhou Yu, Alex Millane, Vikram Ramasamy, Remo Steiner, Sangeeta Subramanian, Clemens Volk, CY Chen, Neel Jawale, Ashwin Varghese Kuruttukulam, Michael A. Lin, Ajay Mandlekar, Karsten Patzwaldt, John Welsh, Huihua Zhao, Fatima Anes, Jean-Francois Lafleche, Nicolas Moënne-Loccoz, Soowan Park, Rob Stepinski, Dirk Van Gelder, Chris Amevor, Jan Carius, Jumyung Chang, Anka He Chen, Pablo de Heras Ciechomski, Gilles Daviet, Mohammad Mohajerani, Julia von Muralt, Viktor Reutskyy, Michael Sauter, Simon Schirm, Eric L. Shi, Pierre Terdiman, Kenny Vilella, Tobias Widmer, Gordon Yeoman, Tiffany Chen, Sergey Grizan, Cathy Li, Lotus Li, Connor Smith, Rafael Wiltz, Kostas Alexis, Yan Chang, David Chu, Linxi "Jim" Fan, Farbod Farshidian, Ankur Handa, Spencer Huang, Marco Hutter, Yashraj Narang, Soha Pouya, Shiwei Sheng, Yuke Zhu, Miles Macklin, Adam Moravanszky, Philipp Reist, Yunrong Guo, David Hoeller, and Gavriel State. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025. URL https://arxiv.org/abs/2511.04831.

[45] Yao Mu, Tianxing Chen, Zanxin Chen, Shijia Peng, Zhiqian Lan, Zeyu Gao, Zhixuan Liang, Qiaojun Yu, Yude Zou, Mingkun Xu, et al. Robotwin: Dual-arm robot benchmark with generative digital twins. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 27649–27660, 2025.

[46] Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. *arXiv preprint arXiv:2603.04356*, 2026.

[47] NVIDIA. Isaac Sim, 2025. URL https://github.com/isaac-sim/IsaacSim.

[48] Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2508.19236*, 2025.

[49] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

[50] Balakumar Sundaralingam, Adithyavairavan Murali, and Stan Birchfield. curobov2: Dynamics-aware motion generation with depth-fused distance fields for high-dof robots, 2026. URL https://arxiv.org/abs/2603.05493.

[51] MotuBrain Team, Chendong Xiang, Fan Bao, Haitian Liu, Hengkai Tan, Hongzhe Bi, James Li, Jiabao Liu, Jingrui Pang, Kiro Jing, et al. Motubrain: An advanced world action model for robot control. *arXiv preprint arXiv:2604.27792*, 2026.

[52] RDT Team. Rdt2: Enabling zero-shot cross-embodiment generalization by scaling up umi data, September 2025. URL https://github.com/thu-ml/RDT2.

[53] Spirit AI Team et al. Spirit-v1. 5: Clean data is the enemy of great robot foundation models. spirit ai blog, 2026.

[54] Marcel Torne, Karl Pertsch, Homer Walke, Kyle Vedder, Suraj Nair, Brian Ichter, Allen Z. Ren, Haohuan Wang, Jiaming Tang, Kyle Stachowicz, Karan Dhabalia, Michael Equi, Quan Vuong, Jost Tobias Springenberg, Sergey Levine, Chelsea Finn, and Danny Driess. Mem: Multi-scale embodied memory for vision language action models, 2026. URL https://arxiv.org/abs/2603.03596.

[55] Hanwen Wang, Weizhi Zhao, Xiangyu Wang, Siyuan Huang, He Lin, Boyuan Zheng, Rongtao Xu, Gang Wang, Yao Mu, He Wang, et al. Dexjoco: A benchmark and toolkit for task-oriented dexterous manipulation on mujoco. *arXiv preprint arXiv:2605.16257*, 2026a.

[56] Kaixuan Wang, Tianxing Chen, Jiawei Liu, Honghao Su, Shaolong Zhu, Minxuan Wang, Zixuan Li, Yue Chen, Huanang Gao, Yusen Qin, et al. Manitwin: Scaling data-generation-ready digital object dataset to 100k. *arXiv preprint arXiv:2603.16866*, 2026b.

[57] Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Zhibin Tang, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, et al. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 2025.

[58] Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Qian Zhu, He Sun, Yong Wang, Shuailei Ma, et al. A pragmatic vla foundation model. *arXiv preprint arXiv:2601.18692*, 2026.

[59] Adina Yakefu, Bin Xie, Chongyang Xu, Enwen Zhang, Erjin Zhou, Fan Jia, Haitao Yang, Haoqiang Fan, Haowei Zhang, Hongyang Peng, et al. Robochallenge: Large-scale real-robot evaluation of embodied policies. *arXiv preprint arXiv:2510.17950*, 2025.

[60] Ganlin Yang, Zhangzheng Tu, Yuqiang Yang, Sitong Mao, Junyi Dong, Tianxing Chen, Jiaqi Peng, Jing Xiong, Jiafei Cao, Jifeng Dai, et al. Eventvla: Event-driven visual evidence memory for long-horizon vision-language-action policies. *arXiv preprint arXiv:2606.20092*, 2026a.

[61] Xuning Yang, Rishit Dagli, Alex Zook, Hugo Hadfield, Ankit Goyal, Stan Birchfield, Fabio Ramos, and Jonathan Tremblay. Robolab: A high-fidelity simulation benchmark for analysis of task generalist policies. *arXiv preprint arXiv:2604.09860*, 2026b.

[62] Yandan Yang, Shuang Zeng, Tong Lin, Xinyuan Chang, Dekang Qi, Junjin Xiao, Haoyun Liu, Ronghan Chen, Yuzhi Chen, Dongjie Huo, et al. Abot-m0: Vla foundation model for robotic manipulation with action manifold learning. *arXiv preprint arXiv:2602.11236*, 2026c.

[63] Angen Ye, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Hengtao Li, Jie Li, Jindi Lv, Jingyu Liu, et al. Gigaworld-policy: An efficient action-centered world–action model. *arXiv preprint arXiv:2603.17240*, 2026a.

[64] Jinhui Ye, Ning Gao, Senqiao Yang, Jinliang Zheng, Zixuan Wang, Yuxin Chen, Pengguang Chen, Yilun Chen, Shu Liu, and Jiaya Jia. Starvla-α: Reducing complexity in vision-language-action systems. *arXiv preprint arXiv:2604.11757*, 2026b.

[65] Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies, 2026c. URL https://arxiv.org/abs/2602.15922.

[66] En Yu, Haoran Lv, Jianjian Sun, Kangheng Lin, Ruitao Zhang, Yukang Shi, Yuyang Chen, Ze Chen, Ziheng Zhang, Fan Jia, et al. Dm0: An embodied-native vision-language-action model towards physical ai. *arXiv preprint arXiv:2602.14974*, 2026a.

[67] Ryan Yu, Pushi Zhang, Starrick Liu, Brae Liu, Miracle Kang, Shalfun Li, Lights Shi, Ellie Ma, Ping Yang, Chris Pan, et al. Wall-oss-0.5 technical report. *arXiv preprint arXiv:2605.30877*, 2026b.

[68] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[69] Haoqi Yuan, Zhixuan Liang, Anzhe Chen, Ye Wang, Haoyang Li, Pei Lin, Yiyang Huang, Zixing Lei, Tong Zhang, Jiazhao Zhang, et al. Qwen-robotmanip technical report: Alignment unlocks scale for robotic manipulation foundation models. *arXiv preprint arXiv:2606.17846*, 2026a.

[70] Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026b.

[71] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations, 2024. URL https://arxiv.org/abs/2403.03954.

[72] He Zhang, Lingzhu Xiang, Haitao Lin, Zeyu Huang, Minghui Wang, Dingyan Zhong, Yubo Dong, Yihao Wu, Yongming Rao, Dongsheng Zhang, et al. Hy-embodied-0.5-vla: From vision-language-action models to a real-world robot learning stack. *arXiv preprint arXiv:2606.14409*, 2026a.

[73] Kaidong Zhang, Jian Zhang, Rongtao Xu, Yu Sun, Shuoshuo Xue, Youpeng Wen, Xiaoyu Guo, Minghao Guo, Weijia Liufu, Liu Zihou, et al. A1: A fully transparent open-source, adaptive and efficient truncated vision-language-action model. *arXiv preprint arXiv:2604.05672*, 2026b.

[74] Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, et al. Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11142–11152, 2025.

[75] Tianle Zhang, Zhihao Yuan, Dafeng Chi, Peidong Liu, Dongwei Li, Kejun Hu, Likui Zhang, Junnan Nie, Ziming Wei, Zengjue Chen, Yili Tang, Jiayi Li, Zhiyuan Xiang, Mingyang Li, Tianci Luo, Hanwen Wan, Ao Li, Linbo Zhai, Zhihao Zhan, Xiaodong Bai, Jiakun Cai, Peng Cao, Kangliang Chen, Siang Chen, Yixiang Dai, Shuai Di, Yicheng Gong, Chenguang Gui, Yucheng Guo, Peng Hao, Qingrong He, Haoyang Huang, Kunrui Huang, Zhixuan Huang, Shibo Jin, Yixiang Jin, Anson Li, Dongjiang Li, Jiawei Li, Ruodai Li, Yihang Li, Yuzhen Li, Jiaming Liang, Fangsheng Liu, Jing Long, Mingxi Luo, Xing Pan, Hui Shen, Xiaomeng Tian, Daming Wang, Song Wang, Junwu Xiong, Hang Xu, Wanting Xu, Zhengcheng Yu, He Zhang, Jiyao Zhang, Lin Zhao, Chen Zhou, Nan Duan, Yuzheng Zhuang, and Liang Lin. Joyai-ra 0.1: A foundation model for robotic autonomy, 2026c. URL https://arxiv.org/abs/2604.20100.

[76] Tianle Zhang, Zhihao Yuan, Dafeng Chi, Peidong Liu, Dongwei Li, Kejun Hu, Likui Zhang, Junnan Nie, Ziming Wei, Zengjue Chen, et al. Joyai-ra 0.1: A foundation model for robotic autonomy. *arXiv preprint arXiv:2604.20100*, 2026d.

[77] Zongzheng Zhang, Jingrui Pang, Zhuo Yang, Kun Li, Minwen Liao, Saining Zhang, Guoxuan Chi, Jinbang Guo, Huan-ang Gao, Modi Shi, et al. Dexora: Open-source vla for high-dof bimanual dexterity. *arXiv preprint arXiv:2605.18722*, 2026e.

[78] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[79] Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. *arXiv preprint arXiv:2510.10274*, 2025.

[80] Bingyang Zhou, Haoyu Zhou, Tianhai Liang, Qiaojun Yu, Siheng Zhao, Yuwei Zeng, Jun Lv, Siyuan Luo, Qiancai Wang, Xinyuan Yu, et al. Clothesnet: An information-rich 3d garment model repository with simulated clothes environment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20428–20438, 2023.

[81] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Kevin Lin, Abhiram Maddukuri, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

Appendix

A. Acknowledgments

A.1. Contributors

RoboDojo-RealEval Infrastructure Contributors. We sincerely thank Tian Nian, Zijian Cai, Kehe Ye, Yukun Liao, Shaolong Zhu, Qiangyu Chen, Jiahao Zhang and Zichun Chen for their valuable support in developing the RoboDojo-RealEval real-world evaluation infrastructure.

Policy Contributors. We sincerely thank the following contributors for providing policy models and supporting their integration and evaluation in RoboDojo: Jun Guo, Zongzheng Zhang, Hongzhe Bi, Jisong Cai, Xiaofeng Wang, Zheng Zhu, Yuhang Tang, Weijie Ke, Mingleyang Li, Ganlin Yang, Shuai Yang, Hengtao Li, Wenxuan Song, Zhangzheng Tu, Kaidong Zhang, Yu Sun, Shuhe Huang, Junliang Guo, Tong Zhang, Yixing Chen, Pengxiang Ding, Rongxu Cui, and Hengkai Tan.

A.2. Evaluation Integrity and Leaderboard Publication Details

Official remote evaluation. Policies must be evaluated through the RoboDojo online evaluation system, either by submitting a deployable policy package or by connecting a remote policy server. Both simulation and real-world evaluations are conducted through the official RoboDojo pipeline using the released task layouts, standardized scene reset procedure, evaluation protocol, and deployment interface. Reported scores are computed by the official evaluation system rather than self-reported by participants.

Repeated simulation evaluation and multi-embodiment real-world evaluation. For simulation, each submitted policy is evaluated under three random seeds to reduce the influence of stochastic initialization and evaluation noise. Participants may submit either three training-seed checkpoints or one checkpoint evaluated under three evaluation seeds. RoboDojo reports the mean and standard deviation across the three simulation runs. For real-world evaluation, each policy is evaluated on all three robot embodiments, including ARX X5, Piper, and Piper X. We report both per-embodiment and aggregate performance to reflect whether a policy generalizes across different collaborative bimanual platforms.

Hidden verification. During official evaluation, each submitted model is also evaluated on a hidden verification set with randomized layouts. This set is used to detect overfitting, hand-tuning, or gaming of the released public layouts. If the success rate on the hidden verification set differs significantly from that on the public evaluation set, the submission is considered invalid and is excluded from the official verified leaderboard.

Open-source artifact for verified publication. Participants who choose to publish results on the official verified leaderboard must release the full evaluation artifact through **XPolicyLab**. This includes the training and deployment code, the evaluated checkpoint, configuration files, and instructions for model loading, inference, deployment, and evaluation under the unified RoboDojo interface. This requirement applies only at the leaderboard publication stage, not during private evaluation. Releasing the exact artifact used for evaluation enables community inspection and independent reproduction.

B. Comparison with Existing Benchmarks

We compare RoboDojo with existing robot manipulation benchmarks in Table 8. The comparison highlights RoboDojo's distinctive design in terms of benchmark scale, sim-and-real evaluation, capability coverage, task diversity, real-world reproducibility, and remote evaluation support. Compared with prior benchmarks that are often specialized to either simulation or real-world evaluation, RoboDojo provides a unified benchmark suite that connects rapid simulation-based diagnosis with standardized real-world validation.

C. Real-World Evaluation Stability

We evaluate $\pi_{0.5}$, GalaxeaVLA, and InternVLA-A1 over three independent RoboDojo-RealEval runs. Each cell reports the standard deviation of success rate and score across R1–R3, with success-rate deviations in percentage points. The "Overall" row gives the standard deviation of the average performance across runs.

Table 8: **Comparison between RoboDojo and other benchmarks.** "Skill" is the number of operation primitives covered; "Num. Policies" is the number of baseline policies the benchmark integrates/evaluates. N/R indicates not reported. HP indicates heterogeneous parallelization.

Benchmark	Platform	Skill	HP	Bimanual	Tasks	Data Source	Num. Policies	Sim-and-Real
RoboTwin 1.0 (Mu et al., 2025)	SAPIEN	7	×	✓	15	Teleop+DataGen	4	×
RoboTwin 2.0 (Chen et al., 2025a)	SAPIEN	13	×	✓	50	DataGen	11	×
RMBench (Chen et al., 2026b)	SAPIEN	3	×	✓	9	Teleop+DataGen	5	×
AutoBio (Lan et al., 2025)	MuJoCo	3	×	✓	16	DataGen	2	×
DexJoCo (Wang et al., 2026a)	MuJoCo	12	×	✓	11	Teleop	4	×
LIBERO (Liu et al., 2023)	MuJoCo	6	×	×	130	Teleop	3	×
CALVIN (Mees et al., 2022)	PyBullet	10	×	×	34	Teleop	1	×
RLBench (James et al., 2019)	CoppeliaSim	18	×	×	100	DataGen	N/R	×
RoboCasa365 (Nasiriany et al., 2026)	MuJoCo	8	×	×	365	Teleop+DataGen	4	×
SimplerEnv (Li et al., 2024b)	SAPIEN	4	×	×	8	Teleop	4	×
VLABench (Zhang et al., 2025)	MuJoCo	10	×	×	100	DataGen	3	×
BEHAVIOR-1K (Li et al., 2023)	Isaac Sim	19	×	✓	1000	Teleop	6	×
RoboSuite (Zhu et al., 2020)	MuJoCo	9	×	✓	9	Teleop	1	×
Meta-World (Yu et al., 2020)	MuJoCo	13	×	×	50	DataGen	7	×
EBench (Gao et al., 2026)	Isaac Sim	11	×	✓	26	Teleop+DataGen	4	×
RoboLab (Yang et al., 2026b)	Isaac Lab	4	×	×	120	Teleop	5	×
ManipulationNet (Chen et al., 2026c)	Real Robot	5	×	✓	5	N/R	N/R	×
RoboArena (Atreya et al., 2025)	Real Robot	11	×	×	N/R	N/R	7	×
RoboChallenge (Yakefu et al., 2025)	Real Robot	13	×	✓	30	Teleop	5	×
RoboDojo (ours)	Isaac Sim	24	✓	✓	60	Teleop+DataGen	35+	✓

D. Simulation Training and Evaluation Details

D.1. Simulation Training Data Details

Table 10 summarizes the simulation training data used in RoboDojo. The training set contains 35 task directories and 3,500 trajectories, totaling 1,859,602 frames, corresponding to 20.66 hours of bimanual manipulation data recorded at 25 Hz. Each trajectory provides synchronized RGB-D observations from a head-mounted camera and two wrist cameras at a resolution of 640×480. Depth images are normalized and stored as integer values in millimeters. A third-view RGB video is also recorded for preview and visualization. The dataset further includes robot states and next-frame actions, covering end-effector poses, gripper states, and joint positions.

Demonstrations are collected through either automated trajectory synthesis or VR-based teleoperation, depending on task structure and execution difficulty. For tasks in the Generalization, Memory, Long-Horizon, and Precision dimensions, we collect 100 trajectories per task. The Open dimension is reserved for evaluation only and does not include task-specific training demonstrations, since it is designed to assess skill recombination and transfer to unseen task specifications rather than imitation of task-specific demonstrations.

For Generalization tasks, training trajectories are collected under the normal setting without domain randomization. To reduce overfitting to a single visual configuration during supervised fine-tuning, we additionally include one auxiliary DLC task with 100 domain-randomized trajectories. These trajectories include diverse backgrounds, lighting conditions, and clutter layouts, broadening visual exposure without leaking task-specific solutions from the evaluation tasks.

D.2. Simulation Evaluation Details

For each simulation task, the maximum interaction horizon is determined from the demonstration data. Specifically, we compute the 90th percentile of task-specific trajectory lengths and set the evaluation horizon to $1.2\times$ this value. For short tasks generated by automated trajectory synthesis, we use a larger multiplier of $1.5\times$ to avoid premature termination caused by small trajectory-length variations. This setting provides sufficient execution time while keeping the evaluation horizon consistent with the demonstrated task complexity.

Table 9: **Full per-task real-world evaluation stability across repeated runs.**

Task	$\pi_{0.5}$		GalaxeaVLA		InternVLA-A1	
	Std. SR (pp)	Std. Score	Std. SR (pp)	Std. Score	Std. SR (pp)	Std. Score
`connect_charger`	0.0	0.0	0.0	0.0	0.0	0.0
`stack_bowls`	5.8	4.0	15.3	16.6	5.8	3.5
`stack_and_cover_blocks`	5.8	5.8	0.0	0.0	0.0	0.0
`fill_pen_holder`	10.0	1.2	5.8	11.7	0.0	3.5
`stand_up_bottles`	5.8	6.6	0.0	7.8	15.3	9.8
`put_objects_into_basket`	5.8	5.8	5.8	5.8	5.8	5.8
`hang_mugs`	0.0	1.2	0.0	2.3	0.0	2.3
`pack_objects_into_backpack`	5.8	3.5	0.0	1.7	0.0	6.5
`sweep_blocks`	5.8	5.8	5.8	5.8	0.0	0.0
`disassemble_LEGO`	5.8	4.6	0.0	1.2	0.0	0.0
`classify_objects`	0.0	2.3	0.0	0.0	0.0	0.0
`cap_pen`	0.0	2.0	0.0	2.3	0.0	2.0
`pack_and_pour_fruit`	5.8	3.1	0.0	3.5	0.0	10.0
`store_in_safe`	23.1	12.9	11.5	13.9	17.3	10.4
`make_food`	5.8	5.1	0.0	1.7	0.0	0.0
`insert_tubes`	5.8	6.4	0.0	8.4	0.0	4.6
`cover_blocks`	5.8	4.6	0.0	0.3	0.0	0.0
`make_bread`	0.0	0.0	0.0	0.0	0.0	0.0
Overall	**1.0**	**0.5**	**1.2**	**1.0**	**1.3**	**1.2**

Table 10: **Simulation training data statistics.** The simulation training set contains 1,859,602 frames from 3,500 trajectories, corresponding to 20.66 hours of bimanual manipulation data at 25 Hz. The Open dimension is excluded from the training set and used only to evaluate skill recombination and transfer to unseen task specifications. DLC denotes an auxiliary domain-randomized task used to broaden visual exposure during training.

Category	**Frames**	**Duration**	**Trajectories**	**Avg. Frames / Traj.**	**Avg. Duration / Traj.**
Generalization	592,432	6.58 h	1,200	494	19.75 s
Memory	328,975	3.66 h	600	548	21.93 s
Precision	368,459	4.09 h	800	461	18.42 s
Long-Horizon	504,133	5.60 h	800	630	25.21 s
Open	0	0.00 h	0	–	–
DLC	65,603	0.73 h	100	656	26.24 s
Total	1,859,602	20.66 h	3,500	531	21.25 s

E. Real-World Training and Evaluation Details

E.1. Real-World Training Data Details

Table 11 summarizes the real-world demonstration data used in RoboDojo. We collect demonstrations using a homogeneous leader-follower teleoperation setup, where the leader arm has the same embodiment as the follower robot. For each task, we collect 100 demonstrations recorded by four different operators to improve behavior diversity and reduce operator-specific bias. Across three robot embodiments, including ARX X5, Piper, and Piper X, the dataset contains 1,800 trajectories and 1,611,841 frames, corresponding to 17.91 hours of bimanual manipulation data recorded at 25 Hz. Although each embodiment contributes the same number of trajectories, the total duration differs due to embodiment-specific execution speeds, workspace configurations, and task structures.

Each demonstration includes robot joint states, end-effector poses, language annotations, and synchronized RGB observations from three camera views, including one head camera and two wrist cameras. All videos are recorded at a resolution of 640×480. After collection, all trajectories are manually inspected by multiple reviewers. The filtering criteria include whether the RoboDojo-RealEval platform is properly reset before the trial, whether the operator hesitates excessively during execution, and whether the task is successfully completed. The reset check covers protocol-specific conditions such as flattening the table cover and adjusting lighting to the standard intensity. This filtering process ensures that the demonstrations are task-complete, high-quality, and consistent with the real-world evaluation protocol.

Table 11: **Real-world demonstration statistics.** The real-world dataset contains 1,611,841 frames from 1,800 trajectories across three robot embodiments, corresponding to 17.91 hours of bimanual manipulation data at 25 Hz. Each embodiment contains 600 trajectories from 6 tasks, while the duration differs due to embodiment-specific execution speeds and task structures.

Embodiment	Frames	Duration	Trajectories	Avg. Frames / Traj.	Avg. Duration / Traj.
ARX X5	665,071	7.39 h	600	1,108	44.34 s
Piper	539,737	6.00 h	600	900	35.98 s
Piper X	407,033	4.52 h	600	678	27.14 s
Total	1,611,841	17.91 h	1,800	895	35.82 s

E.2. Real-World Evaluation Details

For each real-world task, each policy is evaluated for 10 trials. Evaluation layouts are collected in advance and replayed before each trial to ensure consistent initial conditions across policies and sessions. When a policy outputs end-effector control commands, we use Pink (Caron et al., 2026) as the robot motion planner to convert end-effector targets into executable robot motions.

The maximum execution horizon for each task is determined from the demonstration videos by multiplying the 90th percentile trajectory length by 1.5. Once the step limit is reached, the trial is automatically terminated and the robot returns to its reset pose. The evaluation manager may also manually stop a trial if the robot exhibits unsafe behavior that could damage the platform, such as hitting the table, colliding with the external frame, or self-collision. All evaluation videos are recorded for scoring.

Each video is scored independently by three evaluators under a double-blind protocol. The scoring considers both final task outcome and intermediate sub-step completion. The final trial score is obtained by averaging the three evaluator scores. To improve transparency and fairness, we release all evaluation videos and corresponding evaluator scores for leaderboard submissions, and allow users to appeal potential scoring errors.

Public evaluation videos and closed-source track. For transparency, RoboDojo releases evaluation videos for models included in the official verified leaderboard. These videos provide qualitative evidence of policy behavior and allow the community to inspect success cases, failure modes, and potential abnormal evaluation behavior. Results without the evaluated checkpoint, implementation, configuration files, and reproducibility instructions are reported only in a separate closed-source track, which is marked as non-official and unverified because the results cannot be independently reproduced or inspected by the community.

F. Simulation Platform Details

This section provides additional implementation details of the RoboDojo simulation platform, including the configuration-driven task setup, digital asset processing, heterogeneous parallel simulation, and demonstration collection pipeline.

F.1. Simulation Platform Foundation

The RoboDojo simulation platform is built upon the MagicSim infrastructure (Lu et al., 2026), whose codebase provides the foundation of our simulation stack. RoboDojo adopts MagicSim as its core simulation infrastructure, inheriting its modular manager-based runtime design and physically based object abstraction system. Specifically, RoboDojo reuses MagicSim's environment backbone for launching simulation, configuring physics and rendering, managing deterministic seeds, and exposing standardized hooks for task-specific scene setup, stepping, action application, and reset. Through MagicSim's manager architecture, key simulation functions such as scene construction, object management, layout generation, camera configuration, data capture, and robot control are modularized into reusable components. RoboDojo further connects this infrastructure with Isaac Lab's vectorized reinforcement learning interface, while preserving MagicSim's manager-driven execution pattern for observation, reward, and termination computation.

RoboDojo also builds on MagicSim's physically grounded object system, which provides unified abstractions for rigid bodies, articulated mechanisms, static geometry, garments, fluids, and scene fixtures. These MagicSim components encapsulate USD asset loading, PhysX-backed simulation setup, material configuration, state querying, and deterministic utility support,

allowing RoboDojo to focus on robotic task construction rather than low-level simulator engineering. We thanks again to MagicSim who provides the foundational simulation architecture, object model, and simulation backend upon which RoboDojo's embodied robotic environments are developed.

F.2. Configuration-Driven Simulation Setup

RoboDojo follows a configuration-first design, where each task is specified by modular YAML files rather than hard-coded simulator scripts. Each configuration defines task-relevant assets, distractor assets, object initialization distributions, robot initialization, camera settings, lighting conditions, background textures, articulation states, success conditions, and evaluation seeds. This design separates task specification from simulator execution, allowing different tasks to share the same runtime while changing task-specific objects, scene layouts, and randomization ranges.

At reset time, the simulator samples scene instances according to the task configuration under deterministic seed control. This allows RoboDojo to generate diverse scene layouts while remaining reproducible across evaluation runs. For example, the same task can be evaluated under different object poses, clutter configurations, lighting conditions, and background textures by changing the evaluation seed, while keeping the sampling process deterministic.

The platform supports rigid, articulated, and deformable assets through a unified scene construction pipeline. Rigid objects are loaded with collision geometry and physical parameters. Articulated objects are instantiated with joint configurations, joint limits, and initial articulation states. Deformable objects are instantiated with estimated material parameters and simulation-ready mesh representations. This unified pipeline enables different asset types to be used under the same task and policy interface.

F.3. Digital Asset Processing and Validation

RoboDojo builds a digital asset library containing rigid, articulated, and deformable objects for task construction and clutter generation. For rigid objects, we collect assets from online repositories and reconstruct additional real-world daily objects using Meshy AI from reference images or text prompts. Reconstructed meshes are converted into simulation-ready USD assets and manually inspected in Isaac Sim before being added to the benchmark library.

Each asset is annotated with semantic and task-level metadata, including language descriptions, object categories, placement annotations, success-checking annotations, and manipulation affordances for automated data generation. These annotations support both task construction and demonstration generation. For example, graspable regions, placement regions, functional parts, and task-specific interaction points are used to ground low-level skills during automated trajectory synthesis and to define task-specific success checks.

For articulated objects, we inspect the kinematic structure, joint configuration, collision geometry, joint limits, and physical stability. For deformable objects, we assign estimated material parameters and realistic textures, and supplement the asset set with selected assets from ClothesNet (Zhou et al., 2023). All assets are validated through simulation rollouts to reduce unstable contacts, incorrect collision geometry, and physically implausible behavior. This validation process improves the physical reliability of contact-rich manipulation tasks and reduces failures caused by asset artifacts rather than policy limitations.

F.4. Heterogeneous Parallelism Implementation

RoboDojo implements heterogeneous parallelism by disabling strict physics replication across cloned environments and allowing each environment to instantiate its own scene modules from seed-controlled configurations. Unlike homogeneous vectorized environments, where parallel instances usually share the same scene template, RoboDojo allows different environments in the same simulator process to contain different object categories, asset geometries, object counts, clutter layouts, articulation structures, and task configurations.

Each environment maintains an independent task state, random seed, robot initialization, observation stream, and success condition, while sharing the same batched stepping and action dispatch interface. This design preserves the efficiency of vectorized execution while allowing parallel evaluation over diverse scene configurations. For example, one environment may evaluate a rigid-object pick-and-place task with clutter distractors, while another may evaluate manipulation of an articulated object with a different joint structure.

For larger-scale evaluation and data collection, RoboDojo partitions episode seeds across multiple GPUs and launches independent simulator processes. Each process runs a subset of tasks and seeds, and the results are aggregated after evaluation.

Combining intra-process heterogeneous environments with inter-process GPU sharding enables scalable evaluation without forcing all parallel environments to share the same scene template. This is especially important for evaluating generalist manipulation policies, since the benchmark should expose policies to diverse object geometries, layouts, and physical configurations rather than repeated rollouts of a single cloned scene.

F.5. Simulation Demonstration Collection

RoboDojo supports two complementary demonstration collection modes: automated trajectory synthesis and VR-based teleoperation. Both modes use the same asset annotation layer, which specifies manipulation-related affordances and task semantics such as graspable regions, placement regions, functional parts, and task-specific interaction points. These annotations support automated skill grounding and task validation.

For automated trajectory synthesis, each task is decomposed into an ordered sequence of manually specified low-level skills. The skill library includes primitives such as `push_up`, `place`, `handover`, `grasp`, `insert`, `open`, `close`, and `stack`. Each skill is grounded in the corresponding object annotations and executed using the cuRobo v2 motion planner (Sundaralingam et al., 2026), which generates physically feasible robot motions. This design enables complex demonstrations to be generated through reusable skill composition.

For tasks that are difficult to synthesize automatically, RoboDojo provides a VR-based teleoperation interface using devices such as Meta Quest and Pico. The system captures the delta 6D pose of the VR controller and maps it to target end-effector motion in simulation, which is solved online by cuRobo v2. For a single 6-DoF ARX X5 arm, spatial motion solving latency can reach approximately 3 ms per planning step. The gripper is controlled through controller buttons, while foot pedals are used to trigger task start and termination, fix robot arms, and provide other efficiency-oriented controls. This interface enables efficient collection of high-quality demonstrations for complex simulation tasks where automated synthesis is unreliable.

G. RoboDojo-RealEval Platform Details

Hardware setup. RoboDojo-RealEval uses a fixed-height white evaluation table with a $1.2\,\mathrm{m} \times 1.2\,\mathrm{m}$ workspace. A replaceable white table cover maintains a consistent visual background and reduces replacement cost caused by surface damage during repeated evaluations. The workspace is enclosed by an external frame of $1.5\,\mathrm{m} \times 1.5\,\mathrm{m} \times 2.1\,\mathrm{m}$. Black curtains are mounted on the left, right, front, and top sides to reduce external illumination, while three linear LED light sources with fixed mounts provide stable lighting over the manipulation workspace. Custom structural components fix the relative geometry between the table, frame, robot arms, and cameras. A bottom connector fixes the relative distance, position, and orientation of the two robot arms, while a steel tube and custom 3D-printed mount fix the head-camera pose. The platform also includes a touchscreen interface and an integrated workstation box for the control computer and cooling system.

Layout replay and evaluation workflow. For each real-world task, we pre-collect reference evaluation layouts. During evaluation, the target layout image is overlaid with transparency on the live observation stream through the web interface. Evaluators compare the current scene with the reference layout and adjust object positions until the live observation aligns with the target image. This provides an intuitive mechanism for replaying initial layouts and reducing reset variation across policies and sessions. In our test with five data collectors, restoring a scene with five objects takes 14 seconds on average.

Safety control and scoring. After scene reset, the evaluator starts policy execution through the touchscreen interface. Since submitted policies may behave unexpectedly due to code errors or poor training, RoboDojo-RealEval provides an emergency stop function. Once triggered, policy control is immediately disabled and the robot slowly returns to a safe reset state. All observation videos are automatically collected and uploaded to the cloud for scoring. Each trajectory is scored by three independent raters who are blind to the identity of the evaluated policy. Samples with large scoring discrepancies are filtered, and the remaining scores are averaged to obtain the final result.

G.1. Hardware Components of RoboDojo-RealEval

Fig. 10 shows the hardware components of the **RoboDojo-RealEval** platform. The platform integrates wrist cameras, a head camera, replaceable collaborative bimanual robot embodiments, a robot and camera support structure, an external frame with controlled lighting and curtains, an integrated workstation box, and a touchscreen-based operation interface. These components jointly standardize sensing, robot placement, illumination, and workspace layout, providing a reproducible hardware foundation for real-world policy evaluation.

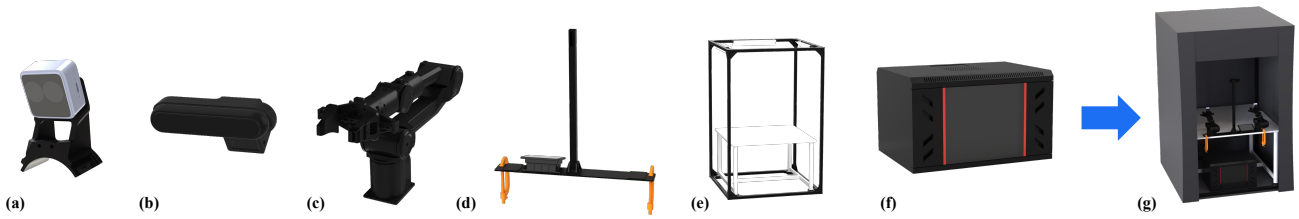


Figure 10: **Hardware components of RoboDojo-RealEval.** The platform consists of (a) two Gemini 305 wrist cameras, (b) one Gemini 335L head camera, (c) replaceable collaborative bimanual robot embodiments including ARX X5, Piper, and Piper X, (d) robot and camera support structure with a touchscreen interface, (e) external frame with LED lighting and curtains, (f) integrated workstation box, and (g) the assembled evaluation platform. These components standardize sensing, robot placement, illumination, and workspace layout for reproducible real-world evaluation.

H. XPolicyLab Design Details

XPolicyLab standardizes the external workflow of robot policy development while allowing each policy to preserve its own internal architecture and implementation. Its design covers three levels: data format, policy interface, and deployment protocol. This separation enables heterogeneous policies, including diffusion policies, VLA models, world-model-based policies, and classical imitation learning baselines, to be trained, debugged, deployed, and evaluated under the same RoboDojo protocol.

Data format and policy development. XPolicyLab converts raw RoboDojo trajectories into policy-specific training formats, such as HDF5 files, LeRobot-style datasets, precomputed language embeddings, and model-specific normalization statistics. At the benchmark level, each trajectory follows a unified observation and action format. Each observation may contain language instructions, multi-view visual inputs, robot states, and metadata. Visual inputs include head, wrist, and third-view cameras, while robot states may include joint positions, end-effector poses, TCP poses, gripper states, and mobile base states. All pose values follow the same convention, represented as position and quaternion in the form $[x, y, z, q_w, q_x, q_y, q_z]$. This unified format separates policy-specific data requirements from the benchmark protocol, allowing different policies to be connected through lightweight adapters and compared under consistent task, robot, action, and seed settings.

Policy interface and debugging. XPolicyLab defines a common policy interface through a standardized `Model` abstraction. Each policy implements a small set of required methods, including `update_obs`, `get_action`, and `reset`. For parallel simulation evaluation, XPolicyLab additionally supports batched interfaces such as `update_obs_batch` and `get_action_batch`. The environment client only calls these standardized functions and does not need to access the internal policy architecture. XPolicyLab also provides an offline debug mode that generates correctly shaped observations and validates returned actions, allowing developers to test model loading, observation parsing, action dimensions, parameter passing, and server-client communication before full simulation or real-world evaluation.

Package organization and deployment protocol. XPolicyLab standardizes the organization of each policy package. A policy package is expected to provide installation, data processing, training, model serving, and evaluation scripts under a consistent structure, such as `install.sh`, `process_data.sh`, `train.sh`, `eval.sh`, `model.py`, `deploy.py`, and `deploy.yml`. The same set of experiment parameters, including dataset name, task name, checkpoint name, robot configuration, action type, and random seed, is used across data processing, training, and evaluation. This convention improves reproducibility and makes it easier to inspect, rerun, and compare experiments across policies.

Communication protocol. XPolicyLab implements policy-environment communication through a lightweight client-server protocol based on **WebSocket** and **MessagePack**. The policy runs as a model server, while the simulator or real robot runs as an environment client. WebSocket provides persistent bidirectional communication for low-latency action queries, and MessagePack serializes observations and actions into a compact binary format that supports images, robot states, nested dictionaries, and action commands. This protocol enables efficient remote communication across offline debugging, heterogeneous parallel simulation, and cloud-based real-world evaluation, while keeping the policy runtime independent of the underlying evaluation environment.

Evaluation loop. The standard evaluation loop is shown in Table 12. At the beginning of each episode, the environment resets the policy state. The environment then repeatedly obtains an observation, sends it to the policy server, requests an action chunk, and executes the returned actions until the episode terminates. The same logic is extended to batched simulation evaluation by maintaining a list of active environment indices and querying the policy with batched observations. This minimal

Table 12: **XPolicyLab unified policy interface and evaluation protocol.** XPolicyLab provides a shared policy-side interface for RoboDojo simulation and real-world evaluation. The same policy implementation can be deployed in single-environment real-world evaluation and batched simulation evaluation, where the batched interface enables heterogeneous parallel execution for faster feedback.

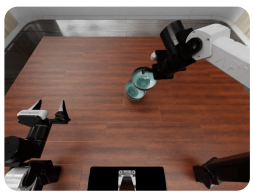
Stage	Single-Environment Deployment	Batched Parallel Deployment
Policy initialization	`policy.reset()` resets the policy state before each episode. This mode is used for real-world evaluation and standard single-task simulation.	`policy.reset()` resets the policy state before evaluating multiple simulation environments in parallel.
Observation update	XPolicyLab receives the latest observation o_t from the RoboDojo environment and updates the policy by `policy.update_obs(`o_t`)`.	XPolicyLab collects active environment indices $\mathcal{I}_t$, receives batched observations $\mathbf{o}_{\mathcal{I}_t}$, and updates the policy by `policy.update_obs_batch(`$\mathbf{o}_{\mathcal{I}_t}$`)`.
Action prediction	The policy predicts an action chunk $\mathbf{a}_{t:t+K}$ through `policy.get_action()`. The same interface is used for simulation and real-world deployment.	The policy predicts batched action chunks $\mathbf{a}_{\mathcal{I}_t,t:t+K}$ through `policy.get_action_batch(`$\mathcal{I}_t$`)`, allowing multiple environments to share one policy service.
Environment execution	RoboDojo executes each action in the chunk through `env.step(`a_k`)` until the episode terminates or the action chunk is exhausted.	RoboDojo executes the k-th action for all active environments through `env.step_batch(`$\mathbf{a}_{\mathcal{I}_t,k}$`,` $\mathcal{I}_t$`)`. Finished environments are removed from the active set.
Evaluation role	Used for remote real-world evaluation in RoboDojo-RealEval, where policies are tested under standardized and reproducible physical conditions.	Used for RoboDojo simulation evaluation, where different tasks, scenes, and processes run concurrently to improve evaluation speed and support rapid policy iteration.

interface supports single-environment evaluation, vectorized simulation, action chunking, and remote real-world deployment.

I. Simulation Benchmark Tasks Details

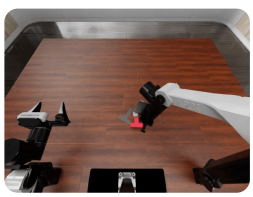
We provide a complete specification for each simulation task, including the task name, the language instructions used for data collection and evaluation, the task description, the source of expert demonstrations, and the task split indicating whether it is used for training or testing. This task-level documentation is intended to make the benchmark transparent and reproducible, and to help users clearly understand the evaluation protocol of each task. More details are available at https://robodojo-benchmark.com/doc.

I.1. Generalization



stack_bowls

Instruction: Stack the three bowls together.
Description: There are three bowls. The robot needs to stack all the bowls together.
Data Source: Teleop **Usage**: Train & Eval

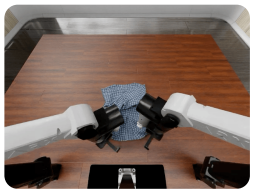


push_T

Instruction: Push the T-shaped block to align it precisely with the gray T-shaped pad.
Description: There is a thin gray T-shaped pad and a T-shaped block. The robot needs to push the T-shaped block until it is precisely aligned with and fitted onto the gray pad.
Data Source: Teleop **Usage**: Train & Eval

pack_objects_into_box

Instruction: Place all the objects into the box with their front sides facing left.
Description: There are several objects on the table and a box. The robot needs to pick up all the objects, place them into the box, and ensure that each object is oriented with its front side facing left.
Data Source: Teleop **Usage**: Train & Eval

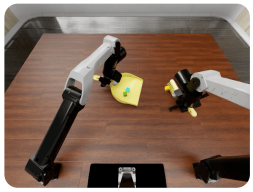


fold_clothes

Instruction: Fold the clothes neatly.
Description: There is a piece of clothing. The robot needs to fold it neatly.
Data Source: AutoGen **Usage**: Train & Eval

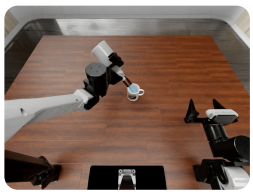
hang_mugs

Instruction: Hang all the mugs on the mug rack.
Description: There are three mugs and one mug rack. The robot needs to pick up each mug and hang all of them on the rack.
Data Source: Teleop **Usage**: Train & Eval



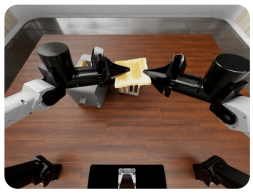
sweep_blocks

Instruction: Pick up the broom, hand it over to the right hand, then use the dustpan to sweep the blocks.
Description: There is a broom and a dustpan on the left side, and the blocks are on the right side. The robot needs to first pick up the broom, hand it over to the right hand, then grasp the dustpan with the left hand, and finally sweep the blocks successfully.
Data Source: Teleop **Usage**: Train & Eval



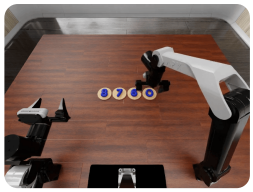
pour_liquid_into_cup

Instruction: Pour the liquid from the bottle into the cup.
Description: There is a bottle containing liquid and a cup. The robot needs to pick up the bottle and pour the liquid into the cup.
Data Source: Teleop **Usage**: Train & Eval



make_toast

Instruction: Pick up two slices of bread, place them into the toaster, and press the lever down.
Description: There is a basket containing multiple slices of bread and a toaster. The robot needs to pick up two slices one by one, place them into the toaster, and then press the lever down to start toasting.
Data Source: Teleop **Usage**: Train & Eval

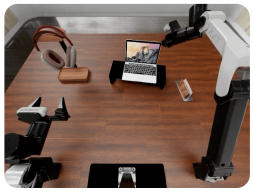


arrange_largest_number

Instruction: Arrange the numbers from left to right to form the largest possible number, and place them on the pad.
Description: There are several number tiles on the table and a pad for placement. The robot needs to determine the order that forms the largest possible number, then place the four numbers on the pad from left to right in that order.
Data Source: AutoGen **Usage**: Train & Eval

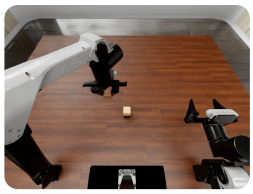
sort_nesting_dolls_by_size

Instruction: Arrange the five nesting dolls in a row from left to right in descending size order.
Description: There are five nesting dolls of different sizes on the table. The robot needs to sort them by size and arrange them in a straight row from left to right, from largest to smallest.
Data Source: AutoGen **Usage**: Train & Eval



store_laptop_and_headphones

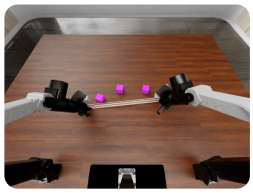
Instruction: Hang the headphones on the headphone stand, close the laptop, then place it into the vertical laptop stand.
Description: There is an open laptop on a laptop stand, a vertical laptop stand, a pair of headphones, and a headphone stand. The laptop opening angle is random but greater than 30 degrees. The robot needs to first hang the headphones on the headphone stand, then close the laptop, pick it up from the stand, and insert it into the vertical laptop stand.
Data Source: Teleop **Usage**: Train & Eval



stack_blocks

Instruction: Stack the three blocks with different textures.
Description: There are three blocks with different textures on the table. The robot needs to pick them up and stack them into a stable pile.
Data Source: AutoGen **Usage**: Train & Eval

I.2. Open

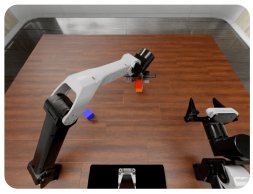


align_blocks

Instruction: Use the set square to push the three blocks into a straight, aligned row, then reset the robot arm.
Description: There is a set square and three blocks. The robot needs to use the set square to push the blocks until they are aligned in parallel in a straight row.
Data Source: null (eval-only) **Usage**: Eval

solve_equation

Instruction: Complete the equation by selecting the correct missing number or operator and placing it on the pad, then reset the robot arm.
Description: There is an arithmetic equation on the table and a pad for the answer. The equation is missing either a number or an operator. The robot needs to choose the correct missing item from the randomly arranged numbers and operators on the table and place it on the pad to complete the equation.
Data Source: null (eval-only) **Usage**: Eval

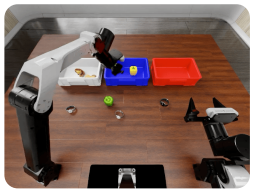


stack_blocks_by_language

Instruction: Stack the three blocks on top of each other in the order of , , and , then reset the robot arm.
Description: There are several colored blocks. The robot needs to understand the language instruction and stack the blocks in the specified color order.
Data Source: null (eval-only) **Usage**: Eval

general_pickup

Instruction: Pick up the by 10 cm.
Description: There are multiple objects. The robot needs to understand the language instruction, identify the target object, and pick it up.
Data Source: null (eval-only) **Usage**: Eval



classify_objects_by_language

Instruction: Put objects into the left basket, objects into the middle basket, and objects into the right basket, then reset the robot arm.

Description: There are three baskets and three categories of unseen objects. The robot needs to understand the language instruction, identify the category of each object, and place the objects into the specified baskets from left to right.

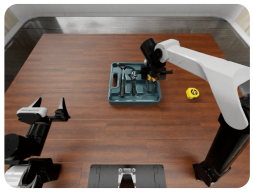
Data Source: null (eval-only) **Usage**: Eval

pick_from_conveyor_by_image

Instruction: Lift the basket more than 8 cm, identify the target object on the conveyor according to the image on the board, pick it up, and place it into the basket.

Description: There is a board displaying an image of the target object, a basket, and a conveyor carrying multiple objects. The robot needs to first lift the basket more than 8 cm, identify the target object on the conveyor based on the image shown on the board, pick up the target object, and place it into the basket.

Data Source: null (eval-only) **Usage**: Eval



store_tools_in_toolbox

Instruction: Place each tool into its matching position in the toolbox, then reset the robot arm.

Description: There are several tools and a toolbox with designated slots. The robot needs to pick up each tool and place it into the corresponding position in the toolbox.

Data Source: null (eval-only) **Usage**: Eval

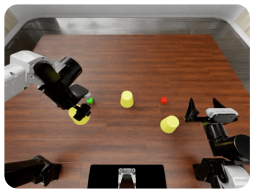
pour_by_language

Instruction: Pour the liquid from the first bottle into the first bowl, from the second bottle into the second bowl, and from the third bottle into the third bowl. Then reset the robot arm.

Description: There are several bottles, cups, and containers on the table. The robot needs to follow the language instruction and pour the liquid from each specified bottle into the corresponding container in the correct order.

Data Source: null (eval-only) **Usage**: Eval

I.3. Memory



cover_blocks

Instruction: Cover the blocks from left to right, remember their colors, then uncover them in the order: red, green, and blue.

Description: There are three covers and three blocks arranged in a random order. The blocks are red, green, and blue. The robot needs to cover the blocks from left to right, remember the color under each cover, and then uncover the blocks in the order of red, green, and blue.

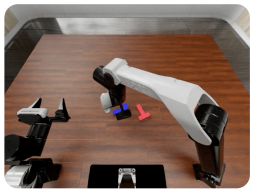
Data Source: AutoGen **Usage**: Train & Eval

match_and_pick_from_conveyor

Instruction: Remember the first object on the conveyor, then pick the matching object when it appears again.

Description: An object first appears on the conveyor and is carried away. The robot needs to remember this object, observe the following objects on the conveyor, and pick the one that matches the first object.

Data Source: Teleop **Usage**: Train & Eval

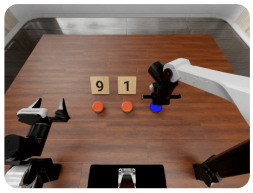


swap_T

Instruction: Pick up the two T-shaped blocks, swap their positions, and place them back with the correct orientations.

Description: There are two T-shaped blocks on the table. The robot needs to grasp them with both hands, swap their positions, and place them back so that each block matches the original pose of the other one, including both position and orientation. This is a memory-based task.

Data Source: AutoGen **Usage**: Train & Eval

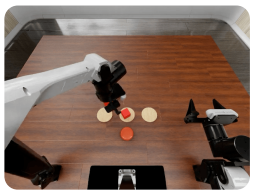


press_by_number

Instruction: Press the two red buttons the required number of times according to the number cards, then press the blue button to confirm.
Description: There are two number cards, two red buttons, and one blue confirmation button. The robot needs to read the numbers, press each red button the corresponding number of times, and then press the blue button to confirm. Pressing the blue button ends the task immediately.
Data Source: AutoGen **Usage**: Train & Eval

imitate_sorting_sequence

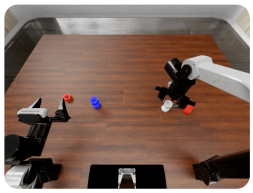
Instruction: Observe the object placement order, remember it, then place the corresponding objects into the basket in the same order.
Description: There are five categories of objects, with five objects on each side. The opposite robot places its objects into the basket on the right side in a certain order. The robot needs to observe and remember this sequence, then place its corresponding objects into the basket in the same order. This is a memory-based imitation task.
Data Source: AutoGen **Usage**: Train & Eval



swap_blocks

Instruction: Swap the two blocks using the empty mat, pressing the button after each move.
Description: There are three mats, two blocks placed on two of the mats, and one button. The robot needs to swap the positions of the two blocks by using the empty mat as a temporary place. After each move, the robot must press the button.
Data Source: AutoGen **Usage**: Train & Eval

I.4. Precision

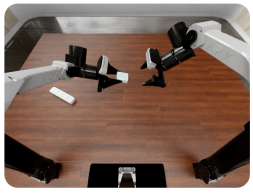


fasten_screws

Instruction: Insert and tighten each screw into the nut of the same color.
Description: There are three screws and three nuts on the table. The screws and nuts come from five possible colors: red, blue, gray, yellow, and purple. In each episode, three different colors are selected, and the robot needs to match each screw with the nut of the same color, insert it, and tighten it. The screw positions vary within a small range, while the nut positions vary within a larger range. If a handover is needed, the robot can first place the screw in the middle and then let the other arm pick it up and fasten it.
Data Source: AutoGen **Usage**: Train & Eval

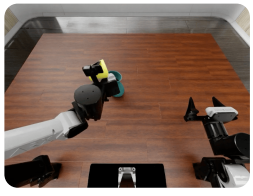
insert_tubes

Instruction: Insert the three tubes into the rack one by one.
Description: There is a tube rack and three tubes. The robot needs to pick up each tube in sequence and insert all tubes into the rack.
Data Source: AutoGen **Usage**: Train & Eval



plug_in_charger

Instruction: Plug the charger into the power strip.
Description: There is a charger plug and a power strip. The robot needs to pick up the charger plug and insert it into the power strip.
Data Source: AutoGen **Usage**: Train & Eval



pour_balls_into_vase

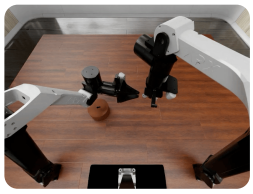
Instruction: Pour all the balls from the cup into the vase.
Description: There is a cup containing many small balls and a vase. The robot needs to pick up the cup and pour all the balls into the vase.
Data Source: Teleop **Usage**: Train & Eval

play_Xylophone

Instruction: Pick up the mallet and strike all xylophone keys from left to right.
Description: There is a xylophone and a mallet. The robot needs to pick up the mallet with one hand and strike all the xylophone keys from left to right.
Data Source: AutoGen **Usage**: Train & Eval

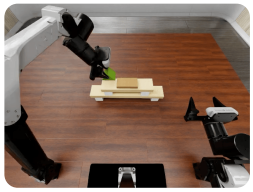
deposit_coin

Instruction: Pick up the coin from the holder and insert it precisely into the coin bank.
Description: There is a coin placed on a holder and a coin bank. The robot needs to pick up the coin and accurately insert it into the slot of the coin bank.
Data Source: AutoGen **Usage**: Train & Eval



insert_key

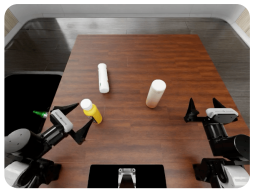
Instruction: Pick up the thick card, hand it over to the other hand to adjust its pose, then insert it into the card slot.
Description: There is a thick card and a card slot on the table. The robot needs to pick up the card, hand it over to the other hand for pose adjustment, and then insert it accurately into the card slot.
Data Source: AutoGen **Usage**: Train & Eval



build_tower

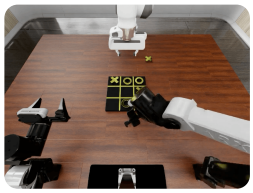
Instruction: Build a tower using the wooden blocks and wooden boards.
Description: There are wooden blocks and wooden boards on the table. The robot needs to use them to build a stable tower structure.
Data Source: AutoGen **Usage**: Train & Eval

I.5. Long-Horizon



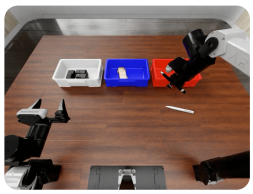
put_bottles_into_dustbin

Instruction: Pick up the bottles and throw them into the dustbin, using handover when needed.
Description: There are four bottles on the table, and they may be either standing or lying down. A dustbin is placed beside the table. The robot needs to pick up the bottles and throw them into the dustbin. Because of the shifted table layout, bottles on the right side require a handover between the two hands before being discarded.
Data Source: Teleop **Usage**: Train & Eval



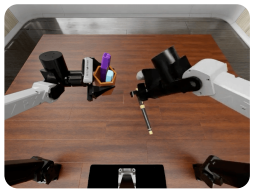
play_tic_tac_toe

Instruction: Play tic-tac-toe as the first player and fill the board with the opponent.
Description: There is a 3-by-3 tic-tac-toe board. The robot plays as the first player, while the opponent follows a random strategy. The robot and the opponent take turns placing their marks until the board is filled.
Data Source: AutoGen **Usage**: Train & Eval



classify_objects

Instruction: Sort the objects by category into the three baskets.
Description: There are three categories of objects and three baskets. The robot needs to group the objects by category and place each category into a separate basket. Any basket can be used for any category, as long as objects of the same category are placed together.
Data Source: Teleop **Usage**: Train & Eval



fill_pen_holder

Instruction: Hold the pen holder with one hand, place all pens into it with the other hand, then put it back down.

Description: There is a pen holder and several pens. The robot needs to grasp the pen holder with one hand, use the other hand to place the pens into the holder one by one, and finally put the filled pen holder back on the table.

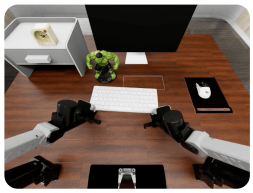
Data Source: Teleop **Usage**: Train & Eval

fill_egg_holder

Instruction: Place the four eggs from the basket into the egg holder, then close the lid.

Description: There is a woven basket containing four eggs and an egg holder. The robot needs to pick up the eggs one by one, place all four into the egg holder, and then close the lid.

Data Source: Teleop **Usage**: Train & Eval



organize_table

Instruction: Place the mouse on the mouse pad, push the keyboard into the frame, put the figurine on the stand, place the alarm clock on the drawer, then open the drawer and put all remaining miscellaneous items inside.

Description: There are a computer, a keyboard, a mouse, an alarm clock, a cartoon figurine, three miscellaneous items, and a drawer. The robot needs to organize the table by placing the mouse on the mouse pad, pushing the keyboard into the frame, putting the figurine on the stand, placing the alarm clock on top of the drawer, opening the drawer, and putting all remaining miscellaneous items into it.

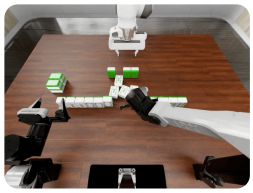
Data Source: Teleop **Usage**: Train & Eval

play_stacking_toy

Instruction: Place all stacking toy pieces onto the correct pegs.

Description: There is a stacking toy with four pegs and four types of pieces. The numbers of pieces in the four types are 4, 3, 2, and 1, and each peg matches one type. The robot needs to place all pieces onto their corresponding pegs correctly.

Data Source: AutoGen **Usage**: Train & Eval



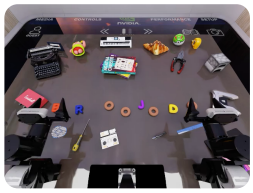
make_kong

Instruction: Wait for the opponent to discard a tile, then declare a kong with the matching tiles.

Description: There is a Mahjong setup. The opponent first pushes out a tile. The robot needs to observe the discarded tile, identify the matching tiles on its side, and perform a valid kong action. The setup guarantees that a kong is possible.

Data Source: AutoGen **Usage**: Train & Eval

I.6. DLC



dlc

Instruction: Arrange the letters to spell "RoboDojo" in a row.

Description: There are multiple separated letters on a cluttered tabletop with a complex background. The robot needs to identify the letters that form "RoboDojo" and arrange them in the correct order in a straight row.

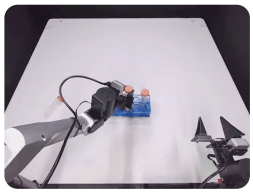
Data Source: null (train-only) **Usage**: Train

J. Real-World Benchmark Tasks Details

We provide a complete specification for each real-world task, including the task name, the language instructions used for data collection and evaluation, the task description, the source of expert demonstrations, and the task split indicating whether it is used for training or testing. This task-level documentation makes the real-world benchmark transparent and reproducible, and helps users understand the standardized evaluation protocol for each physical task. More details are available

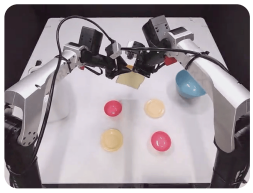
at `RoboDojo-Benchmark.com/doc`.

J.1. ARX X5



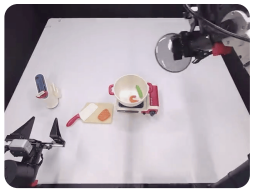
insert_tubes

Instruction: Pick up the test tubes on the table and insert them into the test tube rack.
Description: There are multiple test tubes and a test tube rack on the table. The robot needs to pick up the test tubes one by one and insert them into the slots of the test tube rack.
Data Source: Teleop **Usage**: Train & Eval



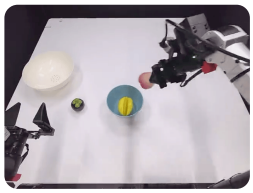
make_bread

Instruction: Pick up the two slices of bread from the bowl and place them into the toaster, then place the two small bowls on the plate.
Description: There are two slices of bread in a bowl, a toaster, two small bowls, and a plate on the table. The robot needs to pick up the bread slices and place them into the toaster, then pick up the two small bowls and place them onto the plate.
Data Source: Teleop **Usage**: Train & Eval



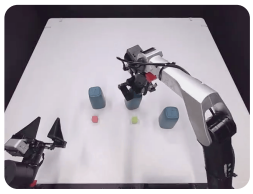
make_food

Instruction: Take the cutting board out and place it on the table. Then place the steak and the knife on the cutting board, put the vegetables and shrimp into the pot, place the pot on the stove, and finally cover it with the lid.
Description: There is a cutting board, a steak, a knife, vegetables, shrimp, a pot, a stove, and a lid in the scene. The robot needs to place the cutting board on the table, arrange the steak and knife on it, put the vegetables and shrimp into the pot, move the pot onto the stove, and finally cover the pot with the lid.
Data Source: Teleop **Usage**: Train & Eval



pack_and_pour_fruit

Instruction: Place all the fruits into the blue bowl, then pour the fruits from the blue bowl into the large white bowl.
Description: There are several fruits, a blue bowl, and a large white bowl on the table. The robot needs to pick up all the fruits and place them into the blue bowl, then transfer them by pouring the fruits from the blue bowl into the large white bowl.
Data Source: Teleop **Usage**: Train & Eval



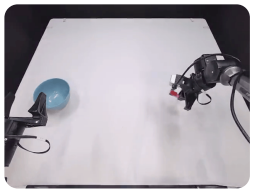
cover_blocks

Instruction: Cover the three blocks from left to right, then uncover them again in the order of red, green, and blue.
Description: There are three lids and three colored blocks on the table: red, green, and blue. The robot needs to first cover the blocks from left to right with the lids, memorize the correspondence between the lids and the block colors, and then uncover them again in the order of red, green, and blue.
Data Source: Teleop **Usage**: Train & Eval

store_in_safe

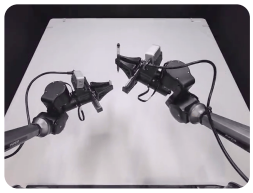
Instruction: Place all the objects into the safe, then close the safe.
Description: There are multiple objects on the table and a safe nearby. The robot needs to pick up all the objects one by one, place them into the safe, and then close the safe.
Data Source: Teleop **Usage**: Train & Eval

J.2. Piper X



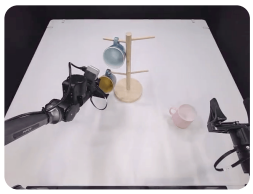
disassemble_LEGO

Instruction: Take apart the assembled LEGO structure and place all the pieces into the bowl.
Description: There is an assembled LEGO structure on the table and a bowl nearby. The robot needs to disassemble the LEGO structure and place all the separated pieces into the bowl.
Data Source: Teleop **Usage**: Train & Eval



cap_pen

Instruction: Pick up the pen and the pen cap, then put the cap on the pen.
Description: There is a pen and a pen cap on the table. The robot needs to pick up both the pen and the cap, align them properly, and put the cap onto the pen.
Data Source: Teleop **Usage**: Train & Eval

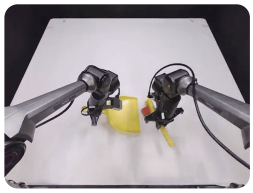


hang_mugs

Instruction: Hang the mugs on the mug rack.
Description: There are several mugs and a mug rack on the table. The robot needs to pick up each mug and hang it on the mug rack.
Data Source: Teleop **Usage**: Train & Eval

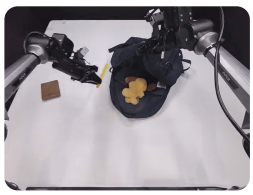
classify_objects

Instruction: Classify the two types of objects on the table and place them into the two baskets.
Description: There are two categories of objects and two baskets on the table. The robot needs to identify the category of each object, classify them accordingly, and place each type into the correct basket.
Data Source: Teleop **Usage**: Train & Eval



sweep_blocks

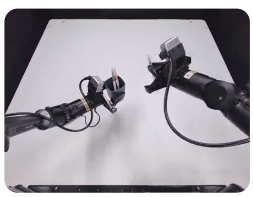
Instruction: Pick up the broom, hand it over to the right hand, then use the dustpan to sweep the blocks.
Description: There is a broom and a dustpan on the left side, and the blocks are on the right side. The robot needs to first pick up the broom, hand it over to the right hand, then grasp the dustpan with the left hand, and finally sweep the blocks successfully.
Data Source: Teleop **Usage**: Train & Eval



pack_objects_into_backpack

Instruction: Place all the objects on the table into the backpack.
Description: There are multiple objects on the table and a backpack nearby. The robot needs to pick up all the objects one by one and place them into the backpack.
Data Source: Teleop **Usage**: Train & Eval

J.3. Piper

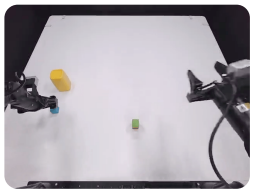


fill_pen_holder

Instruction: Pick up the pen holder and place all the pens into it.
Description: There is a pen holder and several pens on the table. The robot needs to pick up the pen holder and place all the pens into it one by one.
Data Source: Teleop **Usage**: Train & Eval

put_objects_into_basket

Instruction: Place all the objects on the table into the basket.
Description: There are multiple objects on the table and a basket nearby. The robot needs to pick up all the objects one by one and place them into the basket.
Data Source: Teleop **Usage**: Train & Eval

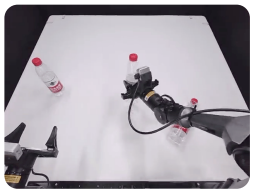


stack_and_cover_blocks

Instruction: Stack the blocks on the table, then cover them with the cup.
Description: There are several blocks and a cup on the table. The robot needs to stack the blocks into a pile and then place the cup over them to cover the stacked blocks.
Data Source: Teleop **Usage**: Train & Eval

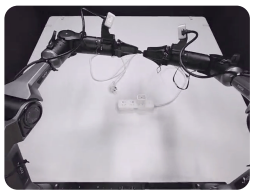
stack_bowls

Instruction: Stack the bowls on the table.
Description: There are several bowls on the table. The robot needs to pick them up one by one and stack them neatly.
Data Source: Teleop **Usage**: Train & Eval



stand_up_bottles

Instruction: Stand the bottle upright.
Description: There is a bottle lying on the table. The robot needs to pick it up and place it upright on the table.
Data Source: Teleop **Usage**: Train & Eval



insert_charger

Instruction: Insert the charger plug into the power strip, then connect the charging cable to the plug.
Description: There is a charger plug, a charging cable, and a power strip on the table. The robot needs to insert the charger plug into the power strip and then connect the charging cable to the plug.
Data Source: Teleop **Usage**: Train & Eval

K. Policy Training Details

In this section, we provide the training settings for all evaluated policies in RoboDojo. For each policy, we report the initialization checkpoint, training data, batch size, number of training steps, and other model-specific configurations when applicable. These details are included to improve reproducibility and to clarify how each policy is trained for simulation and real-world evaluation.

Hy-Embodied-0.5-VLA. We fine-tune Hy-Embodied-0.5-VLA from the foundation checkpoint hosted at `https://huggingface.co/tencent/Hy-Embodied-0.5-VLA-UMI`. For simulation experiments, we follow its official open-source evaluation setting on RoboTwin 2.0, training the model with a batch size of 128 for 200K steps and using 6 historical images sampled every 20 steps for memory encoding.

Spatial Forcing. We fine-tune Spatial Forcing from the `gs://openpi-assets/checkpoints/pi05_base` foundation checkpoint. For simulation experiments, the model is trained with a batch size of 256 for 60K steps.

$\pi_{0.5}$. We fine-tune $\pi_{0.5}$ from the `gs://openpi-assets/checkpoints/pi05_base` foundation checkpoint. For simulation experiments, the model is trained with a batch size of 256 for 60K steps. For real-world experiments, we use the same batch size and train the model for 30K steps.

X-VLA. We fine-tune X-VLA from the `2toInf/X-VLA-Pt` foundation checkpoint. For simulation experiments, the model is trained with a batch size of 256 for 100K steps. For real-world experiments, we use the same batch size and train the model for 50K steps.

Xiaomi-Robotics-0. We fine-tune Xiaomi-Robotics-0 from the `Xiaomi-Robotics-0-Pretrain` foundation checkpoint. For simulation experiments, the model is trained with a batch size of 256 for 100K steps. For real-world experiments, we use the same batch size and train the model for 50K steps.

X-WAM. We fine-tune X-WAM from the `Wan2.2-TI2V-5B` foundation checkpoint. For simulation experiments, the model is trained with a batch size of 32 for 40K steps.

GigaWorld-Policy. We fine-tune GigaWorld-Policy from the `Wan2.2-TI2V-5B` foundation checkpoint. For simulation experiments, the model is trained with a batch size of 32. The video optimization stage is trained for 50K steps (1 epoch), followed by the action optimization stage for 250K steps (2 epochs), resulting in 300K total optimization steps (3 epochs).

StarVLA-α. We fine-tune StarVLA-α from the `Qwen3-VL-4B-Instruct` foundation checkpoint. For simulation experiments, the model is trained with a batch size of 128 for 100K steps. For real-world experiments, we use the same batch size and train the model for 30K steps.

GalaxeaVLA. We fine-tune GalaxeaVLA from the `G0Plus_3B-base` foundation checkpoint. For simulation experiments, the model is trained with a batch size of 32 for 4 epochs. For real-world experiments, we use the same batch size and train the model for 4 epochs.

LingBot-VLA. We fine-tune LingBot-VLA from the `lingbot-vla-4b` foundation checkpoint. For simulation experiments, the model is trained with a batch size of 256 for 15K steps.

EventVLA. We fine-tune EventVLA from the `Qwen3-VL-4B-Instruct` foundation checkpoint. For simulation experiments, the model is trained with a batch size of 128 for 150K steps.

Fast-WAM. We fine-tune Fast-WAM from the `Wan2.2-TI2V-5B` foundation checkpoint. For simulation experiments, the model is trained with a batch size of 256 for 20K steps.

AHA-WAM. We fine-tune AHA-WAM from the `Wan2.2-TI2V-5B` foundation checkpoint. For simulation experiments, the model is trained with a batch size of 768 for 25K steps.

π_0. We fine-tune π_0 from the `gs://openpi-assets/checkpoints/pi0_base` foundation checkpoint. For simulation experiments, the model is trained with a batch size of 256 for 60K steps. For real-world experiments, we use the same batch size and train the model for 30K steps.

GR00T-N1.7. We fine-tune GR00T-N1.7 from the `nvidias/GR00T-N1.7-3B` foundation checkpoint. For simulation experiments, the model is trained with a batch size of 640 for 100K steps. For real-world experiments, we use the same batch size and train the model for 50K steps.

InternVLA-A1. We fine-tune InternVLA-A1 from the `InternVLA-A1-3B` foundation checkpoint. For simulation experiments, the model is trained with a batch size of 64 for 60K steps. For real-world experiments, we use the same batch size and train the model for 30K steps.

SmolVLA (Single Task). We fine-tune SmolVLA (Single Task) from the `smolvla_base` foundation checkpoint. For simulation experiments, the model is trained with a batch size of 512 for 100K steps.

GO-1. We fine-tune GO-1 from the `GO-1` foundation checkpoint. For simulation experiments, the model is trained with a batch size of 96 for 4 epochs, corresponding to 77,484 optimization steps.

LDA-1B. We fine-tune LDA-1B from the `LDA-pretrain` foundation checkpoint. For simulation experiments, the model is trained with a batch size of 128 for 300K steps.

MolmoAct2. We fine-tune MolmoAct2 from the `MolmoAct2` foundation checkpoint. For simulation experiments, the model is trained with a batch size of 128 for 100K steps.

ACT (Single Task). We train ACT (Single Task) from scratch. For simulation experiments, the model is trained with a batch size of 128 for 6K steps.

H-RDT. We fine-tune H-RDT from the `pretrain-0618/checkpoint-500000` foundation checkpoint. For simulation experiments, the model is trained with a batch size of 256 for 90K steps.

Spirit v1.5. We fine-tune Spirit v1.5 from the `Spirit-v1.5` foundation checkpoint. For simulation experiments, the model is trained with a batch size of 256 for 50K steps. For real-world experiments, we use the same batch size and train the model for 30K steps.

RDT-1B. We fine-tune RDT-1B from the `rdt-1b` foundation checkpoint. For simulation experiments, the model is trained with a batch size of 256 for 200K steps.

DM0. We fine-tune DM0 from the `DM0-base` foundation checkpoint. For simulation experiments, the model is trained with a batch size of 32 for 100K steps. For real-world experiments, we use the same batch size and train the model for 50K steps.