WorldFailBench: Do World Models Understand Policy Failures?

Anonymous authors
Paper under double-blind review

ABSTRACT

Robot policy evaluation informs deployment decisions and policy improvement, but physical testing and simulator preparation require substantial effort. World models offer a complementary route through closed-loop interaction in generated environments. Recent benchmarks have demonstrated their potential as policy evaluators through outcome-based evaluation (i.e., assessing task success or failure). However, such evaluation does not establish whether world models faithfully simulate how policies fail. Misrepresenting failures can lead to incorrect diagnoses of policy weaknesses and make policies appear ready for deployment even when small scene changes cause them to fail. We therefore ask: do world models faithfully simulate how and at which stages policies fail, and under what conditions they transition from success to failure? To answer this question, we introduce WorldFailBench with three evaluation tasks. Failure reason tests whether models reproduce the observable errors policies make. Failure stage assesses whether failures concentrate at the same stages of full-task execution. Failure boundary examines whether models reproduce the conditions under which policies transition from success to failure under controlled scene changes in simulation. The benchmark covers manipulation tasks in the LIBERO and RoboTwin simulation environments and real-world manipulation tasks performed by the G1 humanoid robot. Experiments across seven world models reveal substantial gaps in their understanding of policy failures, providing guidance for developing more reliable world models for policy evaluation.

1 INTRODUCTION

Generalist robot policies increasingly support diverse manipulation tasks (Team et al., 2024; Kim et al., 2025; Black et al., 2024), with advances in multimodal prompting, bimanual control, and hierarchical reasoning (Jiang et al., 2023; Liu et al., 2025; Wen et al., 2025; Li et al., 2025c). Reliable deployment requires repeated evaluation across tasks and operating conditions. Physical testing occupies robot hardware and requires resets and supervision (Atreya et al., 2025; Zhou et al., 2025), while traditional simulation requires scene construction and adaptation to the target environment (Li et al., 2025a). Learned world models offer a complementary route: predicting future observations conditioned on robot actions (Yang et al., 2024; Wu et al., 2024) enables closed-loop policy evaluation before physical deployment (Quevedo et al., 2026; Guo et al., 2026). To serve as reliable policy evaluators, they should faithfully simulate policy behavior in the target environments.

Recent benchmarks, including WorldArena and WMBench, systematically assess the reliability of world models as policy evaluators in simulated and real-world settings (Shang et al., 2026a; Team et al., 2026b). A common approach is *outcome-based evaluation* (Li et al., 2025b; Quevedo et al., 2026; Guo et al., 2026; Li et al., 2026). Specifically, the same policies are evaluated in a world model and a *reference environment*, either a conventional simulator or a real-world robot setup. Policy success rates in the two environments are compared through cross-policy correlations. Reported agreement with real-world policy performance provides encouraging evidence for the use of world models as policy evaluators (Wang et al., 2026; Jeon et al., 2026).

However, outcome-based evaluation does not establish whether world models faithfully simulate how a policy fails. Figure 1 illustrates three discrepancies in cloth collection and their implications for policy assessment. **(1) Failure reason.** In panel A, reference execution attempts to grasp the cloth but drops it, indicating an action execution failure. The generated rollout instead attempts to

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

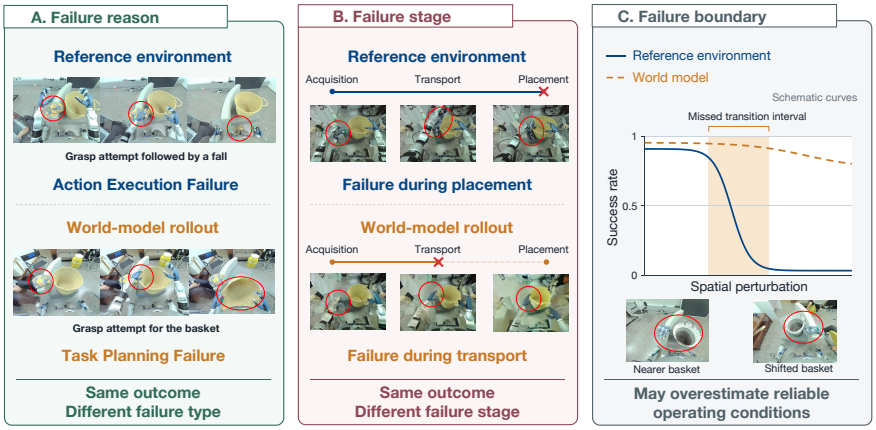


Figure 1: Failure discrepancies in cloth collection. **(A)** Failure reason: dropping the cloth versus grasping the basket. **(B)** Failure stage: reference placement versus model transport. **(C)** Failure boundary: missed transitions under basket displacement.

grasp the basket, choosing the wrong target and indicating a task planning failure. Although both executions fail, this mismatch can misattribute an execution problem to task planning and direct policy improvement toward the wrong component. **(2) Failure stage.** In panel B, reference execution fails during placement, whereas the model predicts failure during transport. This misrepresents where execution breaks down despite agreement on the final outcome. **(3) Failure boundary.** Panel C illustrates how a model can continue to predict high success rates after basket displacement causes reference success to collapse. Missing this success-to-failure transition can overestimate the conditions under which a policy operates reliably. We therefore ask: *do world models faithfully simulate how and at which stages policies fail, and under what conditions they transition from success to failure during closed-loop interaction?*

To answer this question, we introduce WorldFailBench with three complementary evaluation tasks. **Failure reason** asks whether a world model reproduces the kinds of observable errors a policy makes in the reference environment. We compare each policy's type-specific failure frequencies over all evaluation episodes and the correlation of these frequencies across policies in the two environments. **Failure stage** asks whether failures concentrate at the same stages of full-task execution as in the reference environment. We compare each policy's stage-specific failure frequencies over the same episodes and their cross-policy correlation between the two environments. **Failure boundary** asks whether the model captures the conditions under which a policy transitions from success to failure. In simulation, we select pairs of nearby conditions with a drop in reference success rate and test whether the model reproduces the direction and magnitude of that drop. The benchmark spans manipulation tasks in the LIBERO (Liu et al., 2023a) and RoboTwin (Chen et al., 2025) simulation environments and real-world manipulation tasks performed by the G1 humanoid robot. Table 1 compares benchmark designs.

Experiments with seven world models reveal diagnostic gaps that outcome-based evaluation does not expose. On RoboTwin, average errors in predicted planning and execution failure frequencies range from 15.9 to 33.5 percentage points. All seven models show weak stage-specific policy correlations, including the model that best tracks policy success rates. Under scene changes, the models also fail to reliably capture success-to-failure transitions: 95.2% of their responses on the tested RoboTwin boundary pairs understate the magnitude of reference success-rate changes. Our benchmark is publicly available at `https://anonymous.4open.science/r/WORLDFAILBENCH-964A/`. Our contributions are:

- We introduce WorldFailBench, which evaluates world-model fidelity from three complementary perspectives: failure types, failure stages, and failure boundaries.
- We collect reference and model rollouts with failure annotations in LIBERO and RoboTwin simulation and on the G1 robot, and construct controlled boundary pairs in simulation.
- Extensive experiments with seven world models show that outcome-based agreement does not ensure faithful failure diagnosis or boundary prediction, providing guidance for developing more reliable world models for policy evaluation.

Table 1: Comparison of benchmark designs. ✓: dedicated evaluation; △: partial or indirect; ×: not evaluated; —: unverified. †: subtask outcome agreement.

	Protocol	Failure understanding			Environment		Embodiment		
Benchmark	WM loop	Reason	Stage-level	Boundary	Sim.	Real	Single	Dual	Humanoid
WorldArena (Shang et al., 2026a)	✓	×	×	×	✓	×	×	✓	×
WMBench (Team et al., 2026b)	✓	×	✓†	×	✓	✓	—	✓	—
WorldArena 2.0 (Shang et al., 2026b)	✓	×	×	×	✓	✓	✓	✓	×
EWMBench (Yue et al., 2025)	×	△	△	×	×	✓	—	✓	—
WorldSimBench (robot) (Qin et al., 2025)	×	×	△	×	✓	×	✓	×	×
RoboWM-Bench (robot) (Jiang et al., 2026)	×	✓	✓	×	✓	✓	✓	×	×
WorldSimProbe (Co et al., 2026)	×	×	×	△	✓	×	✓	✓	×
KineBench (Liu et al., 2026)	×	×	×	×	✓	×	✓	×	×
WorldEcho (Chen et al., 2026)	×	△	×	×	✓	×	×	✓	×
RoboEval (Wang et al., 2025)	×	✓	✓	×	✓	×	×	✓	×
MiraBench (Yang et al., 2026)	×	✓	×	△	✓	✓	✓	✓	✓
WorldFailBench	✓	✓	✓	✓	✓	✓	✓	✓	✓

2 Related Works

World-model-based policy evaluation and benchmarks. Learned world models have increasingly been adopted as interactive surrogate environments for rolling out embodied policies. Systems such as WorldEval, WorldGym, and dWorldEval deploy action-conditioned video generation to estimate policy performance and preserve ranking orders (Li et al., 2025b; Quevedo et al., 2026; Li et al., 2026), while GigaWorld-1 investigates how world-model architectures affect long-horizon, action-faithful evaluation (Team et al., 2026b). Distinct from standalone evaluation pipelines, WorldArena pioneers unified benchmark protocols for embodied world models, with WorldArena 2.0 systematically expanding this paradigm to visuotactile inputs, interactive policy optimization, and cross-embodiment sim-to-real suites (Shang et al., 2026a;b).

Beyond policy-outcome evaluation, related benchmarks assess complementary aspects of world-model fidelity: WorldSimBench and EWMBench probe perceptual, motion, and semantic consistency (Qin et al., 2025; Yue et al., 2025); RoboWM-Bench tests whether generated interactions translate into executable behaviors (Jiang et al., 2026); MiraBench analyzes reliability and over-optimism under failure-inducing actions (Yang et al., 2026); and RoboEval provides structured, stage-aware policy assessments (Wang et al., 2025). As systematically contrasted in Table 1, while existing benchmarks provide outcome, executability, or recovery signals, they do not jointly assess whether a world model preserves the reference policy's failure mechanism, critical failure stage, and local success-to-failure boundary. WorldFailBench targets this missing evaluator-level fidelity through paired reference/model rollouts and controlled boundary interventions.

Fine-grained robotic failure analysis. Diagnosing failure modes is essential for reliable policy deployment and interpretable robot learning. Prior methods explain manipulation failures via multimodal reasoning (Liu et al., 2023b; Duan et al., 2025), monitor runtime consistency and execution progress (Agia et al., 2025), or provide structured, stage-aware policy assessment (Wang et al., 2025). Concurrently, RoboReward and FailBench audit automated success detectors and reward models (Lee et al., 2026; Navasardyan et al., 2026). While these frameworks inspect the *policy* or the judge within assumed ground-truth physical or simulated environments, WorldFailBench explicitly inverts this lens to evaluate the *world model itself*. By pairing reference physical executions with model rollouts, we treat empirical failure distributions, temporal stage shifts, and local failure boundaries as sensitive probes to benchmark the evaluative fidelity of generative world models.

3 WorldFailBench

WorldFailBench evaluates whether world models preserve policy failure behavior. Figure 2 summarizes the comparison: corresponding closed-loop executions provide failure-type and stage annotations, while controlled scene variations reveal success-to-failure conditions. We next describe the execution protocol, metrics, and benchmark construction.

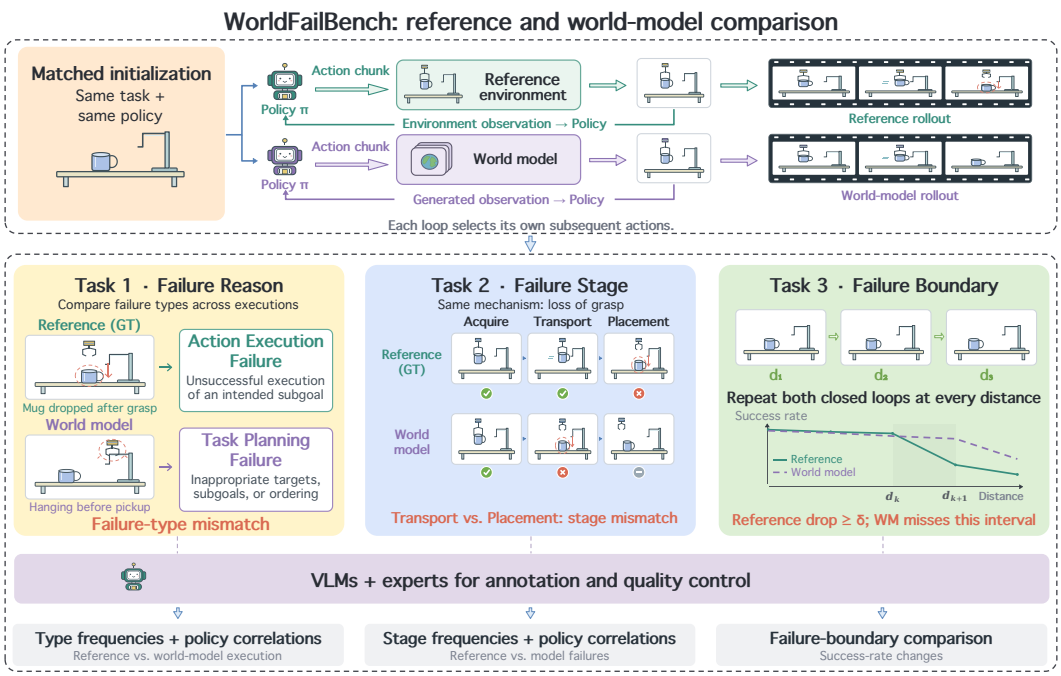


Figure 2: Overview of WorldFailBench: comparing reference and world-model rollouts through failure reason, failure stage, and failure boundary.

3.1 Closed-Loop Evaluation Framework

Closed-loop evaluation tests whether a policy exhibits consistent failure behavior when acting on generated rather than reference observations. At each step, the policy selects an action from its observation history, robot state, and task instruction; the world model predicts the next observation and returns it to the policy. Reference execution follows the same feedback process through a simulator or physical robot (Figure 2). Each loop selects its own subsequent actions, allowing model-generated feedback to affect policy behavior.

To establish a common basis for comparing policy failures, we fix the task instruction, policy, and corresponding initial conditions across environments. Appendix A.3 provides further details of the rollout setup.

3.2 Evaluation Tasks and Metrics

Failure Reason and Failure Stage compare failure frequencies and cross-policy correlations; Failure Boundary tests conditions under which policies transition from success to failure. Within each platform, all world models use the same tasks, policies, and retained reference episodes, with corresponding initial conditions. For task t and policy π, $N_t(\pi) > 0$ counts matched episodes with judged outcomes in each environment, including successes and failures. The environment index r is ref for reference execution or wm for world-model execution. Vertical bars denote set size or absolute value, as appropriate.

3.2.1 Task 1: Failure Reason

Identifying the types of errors a policy makes helps determine which capabilities need improvement. Failure Reason evaluates whether world models reproduce type-specific failure frequencies and their variation across policies in reference execution.

Following REFLECT (Liu et al., 2023b) and Guardian (Pacaud et al., 2027), we distinguish two interpretable failure types: *Task Planning Failure* (inappropriate targets, subgoals, or ordering) and *Action Execution Failure* (unsuccessful execution of an intended subgoal). These types form the TFE set $\mathcal{C}$. Each failed trajectory receives one primary type; unclassifiable confirmed failures are labeled *Indeterminate*. Their episodes remain in $N_t(\pi)$. Appendix B.1 details annotation.

(1) Type Frequency Error (TFE). TFE compares failure-type frequencies for a given policy. For type $c \in \mathcal{C}$, let $n_{r,t}^{\text{type}}(\pi, c)$ count failed episodes carrying that label. Its frequency and task-level error are:

$$f_{r,t}^{\text{type}}(\pi, c) = \frac{n_{r,t}^{\text{type}}(\pi, c)}{N_t(\pi)}, \qquad \text{TFE}_t(\pi) = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \left| f_{\text{ref},t}^{\text{type}}(\pi, c) - f_{\text{wm},t}^{\text{type}}(\pi, c) \right|. \tag{1}$$

Lower TFE indicates closer agreement. Unobserved types have zero frequency; task-level scores are averaged equally for each policy.

(2) Type-specific Policy Correlation (TPC). TPC tests whether a world model preserves cross-policy differences in failure-type frequencies. Following WorldArena (Shang et al., 2026a), we use Pearson correlation across the same J policies $\pi_1, \ldots, \pi_J$. For task t and type c, $\mathbf{f}_{r,t}^{\text{type}}(c) = (f_{r,t}^{\text{type}}(\pi_j, c))_{j=1}^{J}$ lists one frequency per policy in environment r. Let $\mathcal{C}_t \subseteq \mathcal{C}$ contain the types with defined correlations in task t; $|\mathcal{C}_t|$ is their count. We compute each type's correlation and average them within the task:

$$\text{TPC}_t(c) = \text{Pearson}\big(\mathbf{f}_{\text{ref},t}^{\text{type}}(c), \mathbf{f}_{\text{wm},t}^{\text{type}}(c)\big), \quad \text{TPC}_t = \frac{1}{|\mathcal{C}_t|} \sum_{c \in \mathcal{C}_t} \text{TPC}_t(c). \tag{2}$$

Overall TPC averages task scores with at least one defined type correlation equally. Higher TPC indicates stronger positive linear agreement across policies.

3.2.2 Task 2: Failure Stage

Failure Stage evaluates whether world models preserve where decisive failures occur during full-task execution and how their frequencies vary across policies. We divide task t into subgoal-based stages $\mathcal{S}_t$ (Appendix A.1). We label the earliest decisive failure stage when it is identifiable.

(1) Stage Frequency Error (SFE). SFE measures the total absolute discrepancy in a policy's stage-failure frequencies. Let $n_{r,t}^{\text{stage}}(\pi, s)$ count failed episodes labeled $s \in \mathcal{S}_t$. The frequency and task-level error are:

$$f_{r,t}^{\text{stage}}(\pi, s) = \frac{n_{r,t}^{\text{stage}}(\pi, s)}{N_t(\pi)}, \qquad \text{SFE}_t(\pi) = \sum_{s \in \mathcal{S}_t} \left| f_{\text{ref},t}^{\text{stage}}(\pi, s) - f_{\text{wm},t}^{\text{stage}}(\pi, s) \right|. \tag{3}$$

Stage frequencies are computed over all evaluation episodes and reflect both stage reachability and failure after reaching a stage. SFE lies in $[0, 2]$, with lower values indicating closer agreement; task-level scores are averaged equally for each policy.

(2) Stage-specific Policy Correlation (SPC). SPC tests whether a world model preserves cross-policy differences in where decisive failures occur. For stage s, $\mathbf{f}_{r,t}^{\text{stage}}(s) = (f_{r,t}^{\text{stage}}(\pi_j, s))_{j=1}^{J}$ lists frequencies across the same policies. Let $\mathcal{S}_t^{\text{corr}} \subseteq \mathcal{S}_t$ contain the stages with defined correlations in task t; $|\mathcal{S}_t^{\text{corr}}|$ is their count. We compute each stage's Pearson correlation and average them within the task:

$$\text{SPC}_t(s) = \text{Pearson}\big(\mathbf{f}_{\text{ref},t}^{\text{stage}}(s), \mathbf{f}_{\text{wm},t}^{\text{stage}}(s)\big), \quad \text{SPC}_t = \frac{1}{|\mathcal{S}_t^{\text{corr}}|} \sum_{s \in \mathcal{S}_t^{\text{corr}}} \text{SPC}_t(s). \tag{4}$$

Overall SPC averages task scores with at least one defined stage correlation equally. Success-rate correlation applies the same procedure to success rates.

3.2.3 Task 3: Failure Boundary

Identifying success-to-failure conditions can guide deployment and robustness improvements. Failure Boundary tests whether models reproduce these transitions for the same policy across nearby conditions differing in position, orientation, or size (Section 3.3).

For a fixed task, policy, and base scene, $\mathcal{B}$ contains index pairs (i, j) of selected scene conditions x_i and x_j. Let $S_r(x) \in [0, 1]$ be the empirical success rate at condition x in environment r. The change is $\Delta_r(i, j) = S_r(x_i) - S_r(x_j)$, with endpoint order fixed during discovery and shared by both environments.

Table 2: Benchmark composition and reference rollout allocation for failure-reason and failure-stage evaluation. Rollouts per task sum over that task's policies.

Platform	Reference environment	Tasks	Policies / task	Rollouts / task–policy	Rollouts / task	Total rollouts
LIBERO	Sim., single arm	10	3	50	150	1,500
RoboTwin	Sim., bimanual	20	5	50	250	5,000
G1	Real, humanoid	2	4	50	200	400

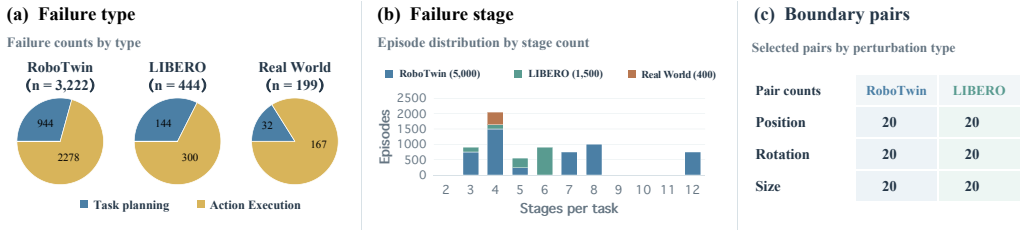


Figure 3: Benchmark data distribution: (a) reference failure counts by type, with n total reference failures per domain; (b) episode counts grouped by the number of named task stages; (c) boundary-pair counts by perturbation type. Real World denotes the real-world tasks on G1.

(1) Order Accuracy (OA). OA measures agreement in nonzero change directions:

$$\mathrm{OA} = \frac{1}{|\mathcal{B}|} \sum_{(i,j)\in\mathcal{B}} \mathbf{1}[\Delta_{\mathrm{ref}}(i,j)\Delta_{\mathrm{wm}}(i,j) > 0]. \tag{5}$$

Here $\mathbf{1}[\cdot]$ equals one for a true condition and zero otherwise; ties score zero. Higher OA is better.

(2) Gap Error (GE). GE measures the error in success-rate changes:

$$\mathrm{GE} = \frac{1}{|\mathcal{B}|} \sum_{(i,j)\in\mathcal{B}} |\Delta_{\mathrm{ref}}(i,j) - \Delta_{\mathrm{wm}}(i,j)|. \tag{6}$$

Lower GE is better. For each factor, we average over pairs within each task–policy–base-scene setting, then equally across settings.

3.3 Benchmark Construction

Benchmark construction comprises task and policy selection, rollout collection, boundary-pair selection, and annotation verification.

Tasks and policies. Table 2 summarizes single-arm LIBERO, bimanual RoboTwin, and real-world tasks on G1. LIBERO uses $\pi_{0.5}$, $\pi_{0.5}$-data-50, and $\pi_{0.5}$-data-20. The data-50 and data-20 variants use 50% and 20% of the policy fine-tuning data, respectively; $\pi_{0.5}$ uses the full dataset (Intelligence et al., 2025). RoboTwin additionally uses LingBot-VLA-v2 (LingBot) (Wu et al., 2026) and Fast-WAM (Yuan et al., 2026). Our real-world evaluation uses the G1 humanoid robot for two tasks: sweeping trash and collecting clothes. Each G1 task uses four task-specific policy versions (Team et al., 2026c), V1–V4. Policy checkpoints remain fixed across environments; Appendix A.1 lists tasks and success criteria.

Failure-reason and failure-stage records. For each task–policy setting, we collect 50 reference rollouts from randomized scenes and corresponding world-model executions using Section 3.1. Both successful and failed rollouts enter annotation and frequency evaluation. Figure 3(a,b) shows reference failure counts by type and episode counts by the number of task stages, respectively.

Boundary pairs. On RoboTwin and LIBERO, we scan one scene factor—object position, orientation, or size—while holding other conditions fixed. Ten reference executions per condition identify adjacent pairs with a success-rate difference of at least $\delta = 0.3$ (30 percentage points). We freeze 60 pairs per platform and their endpoint order, then independently re-estimate reference success using fresh random seeds (Appendix C).

For illustration, a position scan in 1 cm increments may show reference success falling from 90% at a 5 cm offset to 40% at 6 cm, qualifying the pair for selection. Each world model runs the same policy

Table 3: Type Frequency Error (TFE) on RoboTwin (↓). Policy columns average tasks; Mean weights the five policies equally. Bold marks the best world-model values.

Evaluator	$\pi_{0.5}$	$\pi_{0.5}$-data-50	$\pi_{0.5}$-data-20	LingBot	FastWAM	Mean
Ref–Ref	0.059	0.049	0.068	0.057	0.072	0.061
A2World	0.212	0.203	0.217	0.207	0.216	0.211
BWM	0.267	0.230	0.176	0.262	0.244	0.236
Cosmos	0.422	0.272	0.295	0.381	0.304	0.335
Ctrl	**0.153**	0.190	0.141	0.229	**0.142**	0.171
DW05	0.155	**0.166**	**0.136**	**0.164**	0.174	**0.159**
DreamDojo	0.182	0.214	0.303	0.189	0.232	0.224
GE-Sim	0.184	0.174	0.144	0.189	0.227	0.184

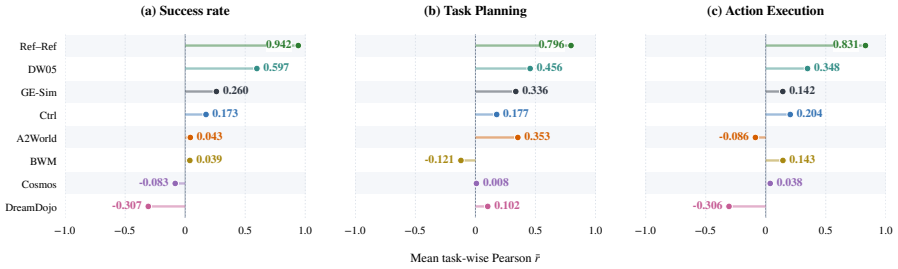


Figure 4: RoboTwin mean within-task Pearson correlations for **(a)** success rate, **(b)** Task Planning, and **(c)** Action Execution. Ref–Ref is the reference-repeatability baseline; each correlation uses five policies.

ten times per frozen endpoint. Pairs are not reselected or reordered after re-estimation. Figure 3(c) gives pair counts by factor.

Annotation and quality control. We use two annotation stages. **(1) Outcome annotation.** Doubao-Seed-2-1-Pro and ChatGPT 6 annotate video outcomes for reference and model rollouts, achieving 90.1% agreement. One expert adjudicates all disagreements. We randomly audit agreed samples totaling 10% of the full dataset, obtaining 97% agreement with human judgments. For simulated reference executions, final outcomes follow native simulator predicates; video labels serve as verification. World-model rollouts and real-world G1 executions use adjudicated video labels. **(2) Failure annotation.** Two experts label the failure type and stage of every confirmed failure, achieving 92% inter-annotator agreement before adjudication. A third annotator resolves all disagreements. Appendix B details annotation rules and outcome verification.

4 Experiments

4.1 Experimental Setup

Evaluated world models. We evaluate Ctrl-World (Ctrl) (Guo et al., 2026), Cosmos-Predict-2.5 (Cosmos) (Ali et al., 2025), BWM (Team et al., 2026a), DreamDojo (Gao et al., 2026), GE-Sim 2.0 (GE-Sim) (Qiu et al., 2026), A2World (Huang et al., 2026) and DW05 (Dexmal, 2026) on RoboTwin. We evaluate the same models except BWM on LIBERO and G1; compatible BWM checkpoints and adaptation code were unavailable. Table 2 summarizes the evaluation scope, and Appendix A.2 gives configurations. The main results examine what failure diagnostics reveal alongside success-rate evaluation on RoboTwin; LIBERO and G1 results provide additional comparisons in Appendices D.4–D.7.
Baseline. Independent executions can differ even when the environment and policy are unchanged. To account for this variation, we rerun the same policies from the same initial scenes in RoboTwin and LIBERO, using the reference environment itself as an ideal simulator. Comparing the new and original executions with the same metrics gives the **Ref–Ref baseline**, which helps assess whether model discrepancies exceed the random variation observed in reference execution (Appendix A.4).

4.2 Failure-Type Understanding

RQ1: Does agreement in policy success rates extend to failure-type fidelity? An evaluator must reproduce the kinds of failures behind the outcomes to support diagnosis of policy weaknesses. We compare success-rate and failure-type correlations across the same five RoboTwin policies (Figure 4), then measure errors in individual policies' failure frequencies using TFE (Table 3).
(1) Success-rate correlation does not establish consistent agreement across failure types. DW05 has the highest success-rate correlation (0.597), but lower correlations for planning (0.456) and

Table 4: Stage Frequency Error (SFE) on RoboTwin (↓). Policy columns average tasks; Mean averages policies within each task, then tasks. Frequencies use all valid paired episodes, including successes. Bold marks the best world-model values.

Evaluator	$\pi_{0.5}$	$\pi_{0.5}$-data-50	$\pi_{0.5}$-data-20	LingBot	FastWAM	Mean
Ref–Ref	0.079	0.104	0.098	0.097	0.093	0.094
A2World	0.691	0.880	0.959	0.613	0.772	0.783
BWM	0.682	0.751	0.808	0.751	0.766	0.751
Cosmos	0.861	0.760	0.871	0.853	0.740	0.817
Ctrl	0.525	**0.611**	**0.679**	0.658	0.691	**0.633**
DW05	**0.521**	0.730	0.818	0.543	**0.663**	0.655
DreamDojo	0.635	0.626	0.815	0.545	0.713	0.667
GE-Sim	0.604	0.823	0.970	**0.517**	0.692	0.721

Table 5: Success-rate correlation and SPC on RoboTwin (↑). Both average five-policy Pearson correlations over tasks; SPC averages stages within each task first. Bold marks the best world-model values.

Evaluator	Success-rate correlation	SPC
Ref–Ref	0.942	0.903
A2World	0.043	−0.061
BWM	0.039	0.045
Cosmos	−0.083	−0.046
Ctrl	0.173	**0.095**
DW05	**0.597**	0.052
DreamDojo	−0.307	−0.123
GE-Sim	0.260	−0.004

Figure 5: FastWAM on `blocks_ranking_size`. Cells show stage failure frequencies (%) over all valid episodes for reference execution and seven world models.



execution failures (0.348). Agreement also differs in direction across types: A2World correlates positively with reference planning failures (0.353) but negatively with execution failures (−0.086), while BWM shows the opposite pattern (−0.121 versus 0.143). The highest aggregate TPC is 0.402 (Table 15). Thus, an evaluator's ability to compare policies depends on which failure type is being assessed; its overall success-rate correlation does not describe this distinction.

(2) World models misestimate how often policies exhibit planning and execution failures, distorting the diagnosis of their weaknesses. Across the seven models, the average absolute error in these failure frequencies is 15.9–33.5 percentage points (Table 3). Generated rollouts therefore give an inaccurate picture of how commonly each kind of failure occurs. For policy evaluation, these frequencies describe the prevalence of specific weaknesses, whereas the overall success rate records only how often tasks are completed.

4.3 Failure-Stage Understanding

RQ2: Do world models preserve where policies fail during full-task execution? We test whether outcome agreement also supports comparisons of failure locations. Table 5 contrasts success-rate correlation with SPC; Table 4 measures stage-frequency errors. We inspect the stage profiles in Figure 5 to explain the discrepancies.

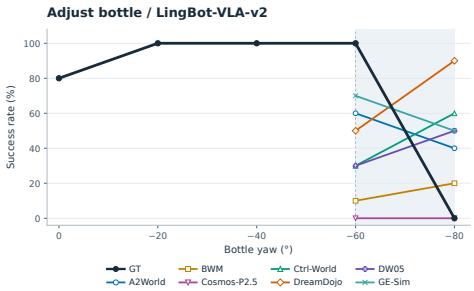
(1) The strongest outcome correlations still accompany weak agreement on where different policies fail. DW05 has the highest success-rate correlation (0.597), yet its SPC is only 0.052; GE-Sim similarly drops from 0.260 to −0.004. Across all seven models, SPC ranges from −0.123 to 0.095. Ref–Ref attains 0.942 for success rate and 0.903 for stages, showing that both comparisons are substantially more reproducible between reference runs. The models' relative strength in outcome evaluation therefore does not carry over to comparing where policies fail during full-task execution.

(2) Generated rollouts misrepresent stage frequencies and can relocate the dominant failure. Mean SFE ranges from 0.633 to 0.817. The FastWAM case in Figure 5 shows what this distortion means: in `blocks_ranking_size`, reference failures concentrate at Grasp small (66%), with 16% at Place, whereas DreamDojo reports 6% and 46%, respectively. This shift from grasping to placement appears across all five policies (Figure 10). The resulting diagnosis attributes the main breakdown in full-task execution to a different stage, information that an overall success rate does not provide.

Table 6: RoboTwin failure-boundary results (OA ↑, GE ↓). World-model scores average settings within each factor. Ref–Ref reports reference repeatability; bold marks the best world model.

Evaluator	Position		Orientation		Size	
	OA ↑	GE ↓	OA ↑	GE ↓	OA ↑	GE ↓
Ref–Ref	1.000	0.005	0.950	0.010	1.000	0.020
A2World	**0.685**	0.515	0.410	0.619	0.455	0.576
BWM	**0.685**	0.529	0.256	0.614	0.364	0.589
Cosmos	0.583	0.587	0.321	0.617	0.394	0.648
Ctrl	0.611	**0.459**	0.359	0.719	**0.758**	**0.500**
DreamDojo	0.602	0.509	0.321	0.677	0.364	0.633
DW05	0.556	0.503	0.308	0.658	0.288	0.636
GE-Sim	0.491	0.481	**0.590**	**0.492**	0.424	0.591

Figure 6: LingBot-VLA-v2 on `adjust_bottle` (pair 032). The reference spans five yaw angles; models connect the highlighted $-60°/-80°$ endpoints.



4.4 Failure-Boundary Understanding

RQ3: Do world models reproduce policy failure boundaries under scene changes? We test whether generated rollouts identify which conditions reduce policy reliability and by how much. Table 6 reports OA and GE for seven world models and the Ref–Ref baseline on 60 RoboTwin boundary pairs spanning position, orientation, and size changes. Each pair holds the task, policy, and base scene fixed; Figure 6 illustrates the resulting errors.

(1) World models misrepresent which scene changes degrade policy reliability and by how much. Even the highest world-model OA for position, orientation, and size is only 68.5%, 59.0%, and 75.8%, respectively (Table 6). Thus, selecting the best model for each factor still leaves substantial disagreement about which nearby condition is less reliable. By comparison, Ref–Ref yields OA of 0.95–1.00 and GE of 0.005–0.020 across the three factors. Predicted changes also tend to be too small: across seven models on the same 60 RoboTwin boundary pairs, 400 of 420 model–pair comparisons (95.2%) underestimate the absolute reference success-rate difference (Figure 11a–c). This compression makes policies appear less sensitive to scene changes than they are. Figure 6 illustrates both errors: on `adjust_bottle`, LingBot-VLA-v2's reference success falls from 100% to 0% as bottle yaw changes from $-60°$ to $-80°$. Four models predict an increase, and one predicts no change. Only A2World and GE-Sim identify the decline, but both predict a decrease of just 20 percentage points. Even a correct direction therefore leaves most of the reliability loss unrepresented in this case. For a policy evaluator, these errors can lead to overly optimistic robustness assessments by obscuring both the conditions that trigger failure and the severity of the resulting reliability loss.

(2) Different scene changes expose distinct weaknesses in world-model policy evaluation. Five of the seven models have their lowest OA on orientation changes, while DW05 and GE-Sim have their lowest OA on size changes (Table 6). Ctrl-World illustrates this dependence: its OA falls from 75.8% for size to 35.9% for orientation, where it also has the highest GE among the seven models (0.719). GE-Sim exhibits the opposite pattern, with OA reaching 59.0% for orientation but only 42.4% for size. Testing size alone would miss Ctrl-World's difficulty with rotation, whereas testing orientation alone would miss GE-Sim's weaker size predictions. An evaluator that captures a policy's sensitivity to one scene factor may therefore misjudge its vulnerability to another. Failure-boundary evaluation makes this limitation explicit: it assesses whether a world model can support judgments about policy robustness under specific scene changes and identifies where its predictions require reference-environment verification before guiding further testing.

5 Discussion and Conclusion

We introduced WorldFailBench to evaluate whether world models preserve policy failure behavior beyond success rates. By comparing generated rollouts with reference executions, it measures agreement in failure types, failure stages, and responses to scene changes near failure boundaries. Across the evaluated models, agreement in policy success rates did not consistently extend to failure diagnosis. Models with stronger outcome correlations could still misrepresent which failures occurred or when they occurred, while generated rollouts often understated the effects of scene changes on success. These findings show why failure fidelity matters when using world models to interpret policy behavior. Our evaluation covers a finite set of policies, tasks, and boundary conditions, which limits the generality of these findings. Future work will extend the evaluation to broader settings and examine whether assessing failure fidelity improves policy training.

REFERENCES

Christopher Agia, Rohan Sinha, Jingyun Yang, Ziang Cao, Rika Antonova, Marco Pavone, and Jeannette Bohg. Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 689–723. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/agia25a.html.

Arslan Ali, Junjie Bai, Maciej Bala, Yogesh Balaji, Aaron Blakeman, Tiffany Cai, Jiaxin Cao, Tianshi Cao, Elizabeth Cha, Yu-Wei Chao, et al. World simulation with video foundation models for physical ai. *arXiv preprint arXiv:2511.00062*, 2025.

Pranav Atreya, Karl Pertsch, Tony Lee, Moo Jin Kim, Arhan Jain, Artur Kuramshin, Cyrus Neary, Edward S. Hu, Kanav Arora, Kirsty Ellis, Luca Macesanu, Matthew Leonard, Meedeum Cho, Ozgur Aslan, Shivin Dass, Tony Wang, Xingfang Yuan, Abhishek Gupta, Dinesh Jayaraman, Glen Berseth, Kostas Daniilidis, Roberto Martín-Martín, Youngwoon Lee, Percy Liang, Chelsea Finn, and Sergey Levine. Roboarena: Distributed real-world evaluation of generalist robot policies. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 336–364. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/atreya25a.html.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Sixiang Chen, Jiaming Liu, Jixian Wu, Yichen Guo, Tinghao Wang, Siyuan Qian, Hao Chen, Jiajun Cao, Jian Tang, and Shanghang Zhang. Do robotic world models really follow actions? diagnosing and aligning action-conditioned generation for policy learning, 2026. URL https://arxiv.org/abs/2608.24885.

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*, 2025.

Peterson Co, Sicheng Hu, Chunxuan Jiao, Hongyang Cheng, Yulin Luo, Yijie Xu, Sixiang Chen, Zhongxia Zhao, Zihao Wang, DaFeng Chi, Peidong Liu, YuTong Chen, Henghua Liu, Zhihao Yuan, Huizhu Jia, Yuzheng Zhuang, Tianle Zhang, Liang Lin, Huajie Tan, and Shanghang Zhang. Worldsimprobe: Diagnosing simulator faithfulness in action-conditioned world models for embodied manipulation, 2026. URL https://arxiv.org/abs/2608.09298.

Dexmal. DW05: A multimodal world model for embodied intelligence. GitHub repository, 2026. URL https://github.com/dexmal/OpenDW.

Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. Aha: A vision-language-model for detecting and reasoning over failures in robotic manipulation. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 45493–45517, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/70a06501001e1820fd1eb9ee821302d2-Paper-Conference.pdf.

Shenyuan Gao, William Liang, Kaiyuan Zheng, Ayaan Malik, Seonghyeon Ye, Sihyun Yu, Wei-Cheng Tseng, Yuzhu Dong, Kaichun Mo, Chen-Hsuan Lin, Qianli Ma, Seungjun Nah, Loic Magne, Jiannan Xiang, Yuqi Xie, Ruijie Zheng, Dantong Niu, You Liang Tan, K.R. Zentner, George Kurian, Suneel Indupuru, Pooya Jannaty, Jinwei Gu, Jun Zhang, Jitendra Malik, Pieter Abbeel, Ming-Yu Liu, Yuke Zhu, Joel Jang, and Linxi Fan. Dreamdojo: A generalist robot world model from large-scale human videos. In *International Conference on Machine Learning (ICML)*, 2026.

Yanjiang Guo, Lucy Shi, Jianyu Chen, and Chelsea Finn. Ctrl-world: A controllable generative world model for robot manipulation. In *International Conference on Learning Representations*, volume 2026, pp. 6121–6138, 2026.

Ze Huang, Jiahui Zhang, Hairuo Liu, Chenxi Zhang, Ran Cheng, and Li Zhang. Learning transferable dynamics priors from action to world modeling, 2026. URL https://arxiv.org/abs/2606.29501.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054.

Byeongguk Jeon, Seonghyeon Ye, JaeHyeok Doo, Sungdong Kim, Minjoon Seo, Hyungmok Son, and Kimin Lee. Roboworld: Fast and reliable neural simulators for generalist robot policy evaluation. *arXiv preprint arXiv:2607.01060*, 2026.

Feng Jiang, Yang Chen, Kyle Xu, Yuchen Liu, Haifeng Wang, Zhenhao Shen, Jasper Lu, Shengze Huang, Yuanfei Wang, Chen Xie, and Ruihai Wu. Robowm-bench: A benchmark for evaluating world models in robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2026.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. VIMA: Robot manipulation with multimodal prompts. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 14975–15022. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/jiang23b.html.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Tony Lee, Andrew Wagenmaker, Karl Pertsch, Percy Liang, Sergey Levine, and Chelsea Finn. Roboreward: General-purpose vision-language reward models for robotics. *arXiv preprint arXiv:2601.00675*, 2026.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Oier Mees, Karl Pertsch, 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. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 3705–3728. PMLR, 06–09 Nov 2025a. URL https://proceedings.mlr.press/v270/li25c.html.

Yaxuan Li, Yichen Zhu, Junjie Wen, Chaomin Shen, and Yi Xu. Worldeval: World model as real-world robot policies evaluator. *arXiv preprint arXiv:2505.19017*, 2025b.

Yaxuan Li, Zhongyi Zhou, Yefei Chen, Yaokai Xue, and Yichen Zhu. dworldeval: Scalable robotic policy evaluation via discrete diffusion world model. *arXiv preprint arXiv:2604.22152*, 2026.

Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Caelan Garrett, Fabio Ramos, Dieter Fox, Anqi Li, Abhishek Gupta, and Ankit Goyal. Hamster: Hierarchical action models for open-world robot manipulation. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 24040–24068, 2025c. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/3bfee3bc6639c36e6e7b058db909f760-Paper-Conference.pdf.

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, 2023a.

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 Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 29982–30009, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/49f80e4d2471ad4f2edf4f5f1ab62339-Paper-Conference.pdf.

Zeyi Liu, Arpit Bahety, and Shuran Song. REFLECT: Summarizing robot experiences for failure explanation and correction. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 3468–3484. PMLR, 2023b. URL https://proceedings.mlr.press/v229/liu23g.html.

Zeyu Liu, Zhangzhe Zhu, Yang Zhang, Chenyou Fan, Chenjia Bai, and Xuelong Li. Kinebench: Benchmarking embodied world models via idm-free kinematic grounding, 2026. URL https://arxiv.org/abs/2607.19876.

Zaruhi Navasardyan, Tatul Danielyan, and Hrant Davtyan. Failbench: How reliable are vlms at judging robot task success? *arXiv preprint arXiv:2609.03611*, 2026.

Paul Pacaud, Ricardo Garcia Pinel, Shizhe Chen, and Cordelia Schmid. Guardian: Detecting robotic manipulation failures with vision-language models. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2027. URL https://arxiv.org/abs/2512.01946. Accepted for publication.

Yiran Qin, Zhelun Shi, Jiwen Yu, Xijun Wang, Enshen Zhou, Lijun Li, Zhenfei Yin, Xihui Liu, Lu Sheng, Jing Shao, Lei Bai, and Ruimao Zhang. Worldsimbench: Towards video generation models as world simulators. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 50338–50362. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/qin25f.html.

Boxiang Qiu, Liliang Chen, Yue Liao, Nan Wang, Lintao Wang, Jiayi Luo, Wenzhi Zhao, Shengcong Chen, Di Chen, Ye Li, et al. Ge-sim 2.0: A roadmap towards comprehensive closed-loop video world simulators for robotic manipulation. *arXiv preprint arXiv:2605.27491*, 2026.

Julian Quevedo, Ansh Kumar Sharma, Yixiang Sun, Varad Suryavanshi, Percy Liang, and Sherry Yang. Worldgym: World model as an environment for policy evaluation. In *International Conference on Learning Representations*, volume 2026, pp. 78932–78957, 2026.

Yu Shang, Zhuohang Li, Yiding Ma, Weikang Su, Xin Jin, Ziyou Wang, Lei Jin, Xin Zhang, Yinzhou Tang, Haisheng Su, et al. Worldarena: A unified benchmark for evaluating perception and functional utility of embodied world models. *AI Open*, 7:208–226, 2026a.

Yu Shang, Yinzhou Tang, Yiding Ma, Zhuohang Li, Lei Jin, Weikang Su, Xin Jin, Zhaolu Wang, Ziyou Wang, Xin Zhang, et al. Worldarena 2.0: Extending embodied world model benchmarking on modality, functionality and platform. *arXiv preprint arXiv:2605.17912*, 2026b.

BWM Team et al. Bwm: A low-cost high-fidelity world simulator for robot learning. *arXiv preprint arXiv:2607.29302*, 2026a.

GigaWorld Team, Angyuan Ma, Boyuan Wang, Bohan Li, Chaojun Ni, Guo Li, Guan Huang, Guosheng Zhao, Hao Li, Hengtao Li, et al. Gigaworld-1: A roadmap to build world models for robot policy evaluation. *arXiv preprint arXiv:2607.02642*, 2026b.

Kairos Team, Fei Wang, Shan You, Qiming Zhang, Tao Huang, Zuoyi Fu, Zhisheng Zheng, Yunlong Xi, Feng Lv, Xiaoming Wu, et al. Kairos: A regret-aware native world-action model stack for physical ai. *arXiv preprint arXiv:2606.16533*, 2026c.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Yi Ru Wang, Carter Ung, Christopher Tan, Grant Tannert, Jiafei Duan, Josephine Li, Anh Le, Rishabh Oswal, Markus Grotz, Wilbert Pumacay, et al. Roboeval: Where robotic manipulation meets structured and scalable evaluation. *arXiv preprint arXiv:2507.00435*, 2025.

Yixuan Wang, Rhythm Syed, Fangyu Wu, Mengchao Zhang, Aykut Onol, Jose Barreiros, Hooshang Nayyeri, Tony Dear, Huan Zhang, and Yunzhu Li. Interactive world simulator for robot policy training and evaluation. *arXiv preprint arXiv:2603.08546*, 2026.

Junjie Wen, Yichen Zhu, Minjie Zhu, Zhibin Tang, Jinming Li, Zhongyi Zhou, Xiaoyu Liu, Chaomin Shen, Yaxin Peng, and Feifei Feng. DiffusionVLA: Scaling robot foundation models via unified diffusion and autoregression. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 66558–66574. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/wen25g.html.

Jialong Wu, Shaofeng Yin, Ningya Feng, Xu He, Dong Li, Jianye Hao, and Mingsheng Long. ivideogpt: Interactive videogpts are scalable world models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 68082–68119. Curran Associates, Inc., 2024. doi: 10.52202/079017-2173. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/7dbb5bfab324e3b86af9bd0df15498dd-Paper-Conference.pdf.

Wei Wu, Fangjing Wang, Fan Lu, He Sun, Shi Liu, Yunnan Wang, Yibin Yan, Yong Wang, Shuailei Ma, Xinyang Wang, et al. From foundation to application: Improving vla models in practice. *arXiv preprint arXiv:2607.06403*, 2026.

Sherry Yang, Yilun Du, Seyed Ghasemipour, Jonathan Tompson, Leslie Kaelbling, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 45210–45234, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/c4d66eae503694424123b93ac0fbaf17-Paper-Conference.pdf.

Tianzhuo Yang, Zihan Shen, Zirui Mi, Zhaoyi Zhang, Jiayi Zhou, Jiaming Ji, Juntao Dai, Jiawei Chen, Boyuan Chen, and Yaodong Yang. Mirabench: Evaluating action-conditioned reliability in robotic world models. *arXiv preprint arXiv:2605.29360*, 2026.

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*, 2026.

Hu Yue, Siyuan Huang, Yue Liao, Shengcong Chen, Pengfei Zhou, Liliang Chen, Maoqing Yao, and Guanghui Ren. Ewmbench: Evaluating scene, motion, and semantic quality in embodied world models. *arXiv preprint arXiv:2505.09694*, 2025.

Zhiyuan Zhou, Pranav Atreya, You Liang Tan, Karl Pertsch, and Sergey Levine. Autoeval: Autonomous evaluation of generalist robot manipulation policies in the real world. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 1997–2017. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/zhou25a.html.

A TASKS AND EXPERIMENTAL SETTINGS

A.1 TASKS, SUCCESS CRITERIA, AND STAGES

Failure-reason and failure-stage evaluation covers 20 RoboTwin tasks, all ten LIBERO-10 tasks, and two real-world tasks on G1. For each platform, the task descriptions and success criteria are followed by the stage definitions shared across policies and world models.

Each task–policy setting contains 50 reference episodes. Reference execution stops upon success or exhaustion of its action budget.

A.1.1 ROBOTWIN

RoboTwin spans bimanual coordination, spatial arrangement, tool use, articulated-object interaction, and sequential manipulation. Table 7 lists task instructions, native success criteria, and limits on waypoint commands, each potentially spanning multiple physics steps.

Table 7: RoboTwin tasks for failure-reason and failure-stage evaluation under `demo_clean`. Instructions are abbreviated; policies receive the original episode-specific text. Success conditions summarize native checks. Limits count waypoint commands.

Task / capability	Instruction	Native success criterion	Limit
`beat_block_hammer` *Tool use*	Pick up the hammer and strike the block.	Hammer contacts the block, with planar functional-point errors below 0.02 m per axis.	400
`blocks_ranking_rgb` *Color-based arrangement*	Arrange the blocks in red–green–blue order.	Correct left-to-right order and adjacent spacing; both grippers open.	1200
`blocks_ranking_size` *Size-based arrangement*	Arrange the blocks from large to small.	Correct left-to-right size order and adjacent spacing; both grippers open.	1200
`handover_block` *Bimanual transfer*	Transfer the block between arms and place it at the target.	Block reaches the target within position tolerances; right gripper open.	800
`hanging_mug` *Reorientation and hanging*	Reorient the mug and hang it on the rack.	Mug functional point aligns with the rack target and satisfies the height threshold; right gripper open.	900
`move_pillbottle_pad` *Precise placement*	Place the pill bottle on the pad.	Bottle satisfies planar alignment and resting-height tolerances; both grippers open.	400
`open_laptop` *Articulated-object interaction*	Open the laptop.	Hinge reaches 40% of its joint range; acting end effector remains near the designated contact point.	700
`pick_diverse_bottles` *Bimanual grasping*	Pick up one bottle with each arm.	Both bottles reach their respective planar target regions and required heights.	400
`place_a2b_left` *Spatial positioning*	Place object A to the left of object B.	Correct side, planar distance of 0.08–0.20 m, and lateral alignment; both grippers open.	400
`place_bread_skillet` *Coordinated holding and placement*	Hold the skillet and place the bread inside.	Bread and skillet functional point satisfy planar alignment and height constraints.	500
`place_cans_plasticbox` *Multi-object placement*	Place both cans in the plastic box.	Each can is within 0.04 m planar distance of a box target point; both grippers open.	800

Continued on next page

Table 7 (continued)

Task / capability	Instruction	Native success criterion	Limit
`place_dual_shoes` *Oriented placement*	Place both shoes in the box, pointing left.	Both shoes satisfy target position, height, and orientation constraints; both grippers open.	600
`place_empty_cup` *Precise placement*	Place the cup on the coaster.	Functional-point planar distance below 0.035 m and vertical error below 0.015 m; both grippers open.	500
`place_object_scale` *Placement on a support surface*	Place the object on the scale.	Object satisfies planar alignment and height constraints; acting gripper open.	400
`place_phone_stand` *Three-dimensional alignment*	Place the phone on its stand.	Phone and stand functional points satisfy position tolerances; both grippers open.	400
`put_bottles_dustbin` *Sequential manipulation*	Put all three bottles in the dustbin.	All bottles lie within the specified planar region and height interval.	1700
`put_object_cabinet` *Drawer interaction and placement*	Open the drawer and place the object inside.	Object reaches the cabinet target region and height range; acting gripper open.	700
`scan_object` *Bimanual tool alignment*	Hold the object and align the scanner with it.	Scanner-axis alignment and distance constraints hold; both grippers closed.	500
`stack_blocks_three` *Sequential stacking*	Stack the three blocks in order.	Consecutive blocks satisfy the required vertical offsets and position tolerances; both grippers open.	1200
`stack_bowls_three` *Multi-object stacking*	Stack the three bowls.	Height-sorted bowls satisfy adjacent planar alignment and native one-sided height bounds; both grippers open.	1200

Success follows the implemented predicates: `handover_block` checks terminal placement rather than the transfer sequence, and `put_object_cabinet` does not require drawer closure.

Table 8 lists task-specific stage categories. Both `blocks_ranking_rgb` and `blocks_ranking_size` use eight categories: repeated Reach and Place actions share labels, while grasps distinguish the target blocks and transfers distinguish their directions. Slash-separated parallel actions form one stage. Names are translated into English for presentation; annotation retains the source labels.

Table 8: Task-specific RoboTwin stage categories. Repeated Reach and Place actions share labels in `blocks_ranking_rgb` and `blocks_ranking_size`; other repeated actions retain separate sequence positions. Slash-separated parallel actions form one stage.

Task	Stages
`beat_block_hammer`	**1.** Reach; **2.** Grasp; **3.** Transfer; **4.** Strike
`blocks_ranking_rgb`	**1.** Reach; **2.** Grasp red; **3.** Transfer right; **4.** Place; **5.** Grasp green; **6.** Transfer center; **7.** Grasp blue; **8.** Transfer left
`blocks_ranking_size`	**1.** Reach; **2.** Grasp small; **3.** Transfer right; **4.** Place; **5.** Grasp medium; **6.** Transfer center; **7.** Grasp large; **8.** Transfer left
`handover_block`	**1.** Arm 1: Reach; **2.** Arm 1: Grasp; **3.** Arm 1: Transfer; **4.** Arm 2: Reach; **5.** Arm 2: Grasp; **6.** Arm 1: Release; **7.** Arm 2: Transfer; **8.** Arm 2: Place
`hanging_mug`	**1.** Arm 1: Reach; **2.** Arm 1: Grasp; **3.** Arm 1: Place; **4.** Arm 2: Reach; **5.** Arm 2: Grasp; **6.** Arm 2: Transfer; **7.** Arm 2: Place

Continued on next page

Table 8 (continued)

Task	Stages
move_pillbottle_pad	**1.** Reach; **2.** Grasp; **3.** Transfer; **4.** Place
open_laptop	**1.** Reach; **2.** Grasp; **3.** Lift
pick_diverse_ bottles	**1.** Arm 1: Reach / Arm 2: Reach; **2.** Arm 1: Grasp / Arm 2: Grasp; **3.** Arm 1: Transfer / Arm 2: Transfer
place_a2b_left	**1.** Reach; **2.** Grasp A; **3.** Place A
place_bread_skillet	**1.** Arm 1: Reach / Arm 2: Reach; **2.** Arm 1: Grasp / Arm 2: Grasp; **3.** Rotate the skillet; **4.** Transfer the bread; **5.** Place the bread
place_cans_ plasticbox	**1.** Arm 1: Reach / Arm 2: Reach; **2.** Arm 1: Grasp / Arm 2: Grasp; **3.** Arm 1: Transfer; **4.** Arm 1: Place; **5.** Arm 1: Retract; **6.** Arm 2: Transfer; **7.** Arm 2: Place; **8.** Arm 2: Retract
place_dual_shoes	**1.** Arm 1: Reach / Arm 2: Reach; **2.** Arm 1: Grasp / Arm 2: Grasp; **3.** Arm 1: Transfer; **4.** Arm 1: Place; **5.** Arm 1: Retract; **6.** Arm 2: Transfer; **7.** Arm 2: Place
place_empty_cup	**1.** Reach; **2.** Grasp; **3.** Transfer; **4.** Place
place_object_scale	**1.** Reach; **2.** Grasp; **3.** Transfer; **4.** Place
place_phone_stand	**1.** Reach; **2.** Grasp; **3.** Transfer; **4.** Place
put_bottles_dustbin	**1.** Bottle 1: Reach; **2.** Bottle 1: Grasp; **3.** Bottle 1: Transfer; **4.** Bottle 1: Place; **5.** Bottle 2: Reach; **6.** Bottle 2: Grasp; **7.** Bottle 2: Transfer; **8.** Bottle 2: Place; **9.** Bottle 3: Reach; **10.** Bottle 3: Grasp; **11.** Bottle 3: Transfer; **12.** Bottle 3: Place
put_object_cabinet	**1.** Arm 1: Reach; **2.** Arm 1: Grasp; **3.** Arm 2: Reach; **4.** Arm 2: Grasp; **5.** Arm 2: Pull open; **6.** Arm 1: Transfer; **7.** Arm 1: Place
scan_object	**1.** Arms 1 and 2: Reach; **2.** Arms 1 and 2: Grasp; **3.** Arm 1: Transfer; **4.** Arm 2: Transfer
stack_blocks_three	**1.** Block 1: Reach; **2.** Block 1: Grasp; **3.** Block 1: Place; **4.** Block 1: Retract; **5.** Block 2: Reach; **6.** Block 2: Grasp; **7.** Block 2: Place; **8.** Block 2: Retract; **9.** Block 3: Reach; **10.** Block 3: Grasp; **11.** Block 3: Place; **12.** Block 3: Retract
stack_bowls_three	**1.** Bowl 1: Reach; **2.** Bowl 1: Grasp; **3.** Bowl 1: Place; **4.** Bowl 1: Retract; **5.** Bowl 2: Reach; **6.** Bowl 2: Grasp; **7.** Bowl 2: Place; **8.** Bowl 2: Retract; **9.** Bowl 3: Reach; **10.** Bowl 3: Grasp; **11.** Bowl 3: Place; **12.** Bowl 3: Retract

A.1.2 LIBERO-10

LIBERO-10 contains long-horizon, multi-object tasks with composed goals (Table 9). Three $\pi_{0.5}$ data variants give 150 reference episodes per task and 1,500 in total. The $\pi_{0.5}$ variants use 520 policy actions at 20 Hz, after ten settling steps excluded from the budget.

Table 9: Complete LIBERO-10 task set. All listed goal predicates must hold simultaneously. Limits count policy actions and exclude the ten settling steps.

Task / capability	Instruction	Native success criterion	Limit
task_00 *Two-object placement*	Put both the alphabet soup and the tomato sauce in the basket.	Both objects are inside the basket.	520
task_01 *Two-object placement*	Put both the cream cheese box and the butter in the basket.	Both objects are inside the basket.	520
task_02 *State change and placement*	Turn on the stove and put the moka pot on it.	Stove is on; moka pot is on its cooking region.	520

Continued on next page

Table 9 (continued)

Task / capability	Instruction	Native success criterion	Limit
`task_03` *Placement and drawer closure*	Put the black bowl in the bottom drawer of the cabinet and close it.	Bowl is inside the bottom drawer; drawer is closed.	520
`task_04` *Object–target association*	Put the white mug on the left plate and put the yellow and white mug on the right plate.	Each mug is on its designated plate.	520
`task_05` *Compartment placement*	Pick up the book and place it in the back compartment of the caddy.	Book is inside the caddy's rear compartment.	520
`task_06` *Composed spatial goals*	Put the white mug on the plate and put the chocolate pudding to the right of the plate.	Mug is on the plate; pudding is in the designated region to its right.	520
`task_07` *Two-object placement*	Put both the alphabet soup and the cream cheese box in the basket.	Both objects are inside the basket.	520
`task_08` *Multi-object placement*	Put both moka pots on the stove.	Both pots are on the cooking region; stove remains on.	520
`task_09` *Placement and door closure*	Put the yellow and white mug in the microwave and close it.	Mug is inside the microwave's heating region; door is closed.	520

Success uses the native BDDL predicates `In`, `On`, `Close`, and `Turnon`. For `task_08`, the stove is already on at initialization. Table 10 lists the corresponding subgoal sequences.

Table 10: Ordered stages for all LIBERO-10 tasks.

Task	Ordered stages
`task_00`	**1.** Grasp the alphabet soup.; **2.** Move the alphabet soup to the basket.; **3.** Release the alphabet soup and retreat the arm.; **4.** Grasp the tomato sauce.; **5.** Move the tomato sauce to the basket.; **6.** Release the tomato sauce and retreat the arm.
`task_01`	**1.** Grasp the cream cheese box.; **2.** Move the cream cheese box to the basket.; **3.** Release the cream cheese box and retreat the arm.; **4.** Grasp the butter.; **5.** Move the butter to the basket.; **6.** Release the butter and retreat the arm.
`task_02`	**1.** Turn on the stove.; **2.** Grasp the moka pot.; **3.** Move the moka pot to the stove.; **4.** Place the moka pot on the stove, release it, and retreat the arm.
`task_03`	**1.** Open the bottom drawer.; **2.** Grasp the black bowl.; **3.** Move the black bowl into the bottom drawer.; **4.** Release the bowl and retreat the arm.; **5.** Close the bottom drawer and retreat the arm.
`task_04`	**1.** Grasp the white mug.; **2.** Move the white mug to the left plate.; **3.** Release the white mug and retreat the arm.; **4.** Grasp the yellow-and-white mug.; **5.** Move the yellow-and-white mug to the right plate.; **6.** Release the mug and retreat the arm.
`task_05`	**1.** Grasp the book.; **2.** Move the book to the back compartment of the caddy.; **3.** Release the book and retreat the arm.
`task_06`	**1.** Grasp the white mug.; **2.** Move the white mug to the plate.; **3.** Release the mug and retreat the arm.; **4.** Grasp the chocolate pudding.; **5.** Move the chocolate pudding to the right of the plate.; **6.** Release the pudding and retreat the arm.
`task_07`	**1.** Grasp the alphabet soup.; **2.** Move the alphabet soup to the basket.; **3.** Release the soup and retreat the arm.; **4.** Grasp the cream cheese box.; **5.** Move the cream cheese box to the basket.; **6.** Release the box and retreat the arm.

Continued on next page

Table 10 (continued)

Task	Ordered stages
`task_08`	**1.** Grasp the first moka pot.; **2.** Move the first moka pot to the stove.; **3.** Release the first moka pot and retreat the arm.; **4.** Grasp the second moka pot.; **5.** Move the second moka pot to the stove.; **6.** Release the second moka pot and retreat the arm.
`task_09`	**1.** Open the microwave door.; **2.** Grasp the yellow-and-white mug.; **3.** Move the mug into the microwave.; **4.** Release the mug and retreat the arm.; **5.** Close the microwave door and retreat the arm.

A.1.3 Real-World Tasks on G1

Figure 7 gives the scenes, language instructions, and four task-specific stages for sweeping trash and collecting clothes.

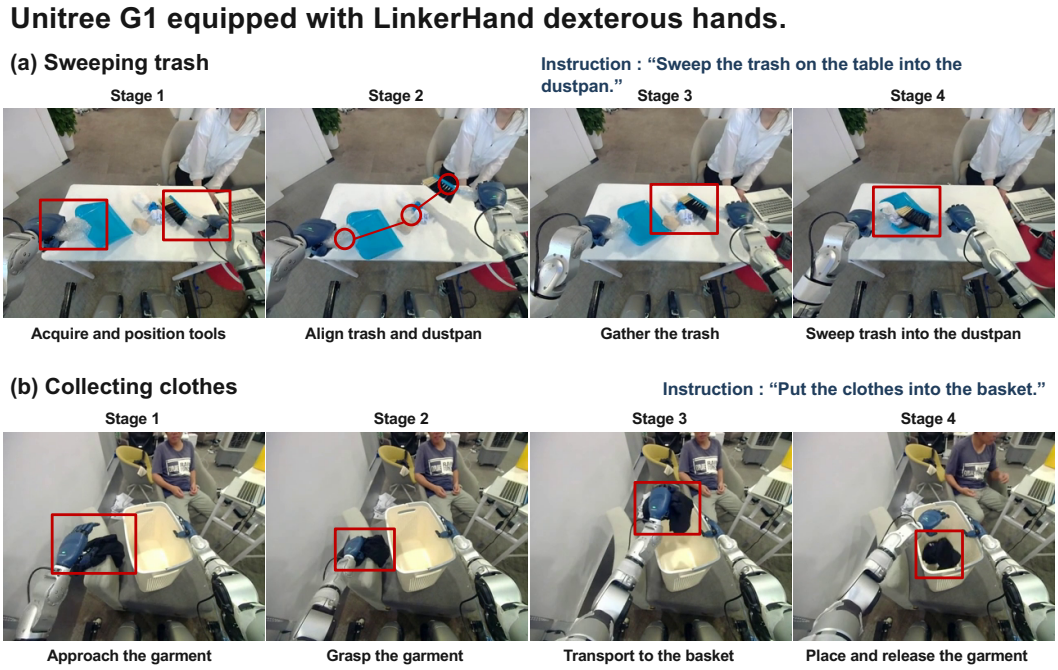


Figure 7: Task scenes, instructions, and stage decompositions for two real-world tasks on G1: (a) sweeping trash and (b) collecting clothes. The robot uses LinkerHand dexterous hands; red annotations highlight relevant objects and alignment.

Sweeping trash. Success requires the main target trash to be swept into the dustpan using the broom. Tiny, visually ambiguous fragments may be tolerated; clearly visible larger pieces remaining outside preclude success.

Collecting clothes. Success requires the target clothing to be placed in the basket and released. Partial draping over the rim is acceptable when the basket supports the clothing; suspension above the basket or continued support by the hand does not satisfy the goal.

A.2 Environment, Model, and Policy Configurations

Evaluation scope. WorldFailBench evaluates complete world-model systems, including their platform adapters, state-feedback rules, and execution schedules. These components vary across models and jointly determine their capabilities for closed-loop policy evaluation. We therefore follow each model's official implementation and inference procedure, adapting observation and action interfaces to the target platform where necessary.

A.2.1 Observation and action interfaces

We evaluate robot policies in LIBERO, RoboTwin, and two real-world manipulation tasks on G1. Each policy receives a task instruction, one observed RGB view, and a platform-specific robot-state vector. When a policy interface includes additional wrist-camera slots, these slots are zero-filled or masked according to that interface. On LIBERO, all three policy variants and all world models use only the main camera view. Table 11 summarizes policy-facing inputs and native action representations.

Table 11: Environment and policy interfaces. Prediction horizon denotes the number of actions returned by a policy call; executed prefix or replanning interval denotes the portion advanced before another call. Action-timeline frequencies and iterative sampling steps are separate quantities. Reference commands are executed through the native platform interfaces described below.

Platform	Environment and policy observations	Policy action space	Policy execution settings
LIBERO	Initial states reconstructed with MuJoCo 3.3.2 and robosuite 1.4.0; 256×256 fixed front-view RGB; 8D robot state; task instruction	7D action: end-effector translation and rotation increments plus a gripper command	The evaluated $\pi_{0.5}$ variants predict 10 actions and execute the first 8 at 20 Hz, replanning every 0.4 s
RoboTwin	`demo_clean` scenes and `seen` instructions; head-camera RGB; 14D robot state; task instruction	14D absolute targets: six joint targets and one gripper target per arm	30 Hz action timeline; $\pi_{0.5}$: prediction horizon 10; LingBot-VLA-v2: reference action chunk 50; FastWAM: reference replanning interval 24 actions and 10 sampling steps
G1	256×448 front-view RGB; 26D robot state; task instruction	26D action	Kairos; four policy versions per task

BWM is evaluated on RoboTwin. Ctrl-World, Cosmos-Predict-2.5, DreamDojo, DW05, A2World, and GE-Sim are evaluated on all three platforms.

RoboTwin reference execution uses the native qpos-to-TOPP interface to execute joint targets. LIBERO gripper outputs are converted to the environment's native commands.

A.2.2 Policy variants

Policy fine-tuning follows the official implementation. The three LIBERO policies are $\pi_{0.5}$-data-20, $\pi_{0.5}$-data-50, and the full-data $\pi_{0.5}$. The full-data variant is named data100 in the training configuration. Their training subsets contain 305, 764, and 1,524 trajectories, respectively, from the 40-task policy-training collection, with a shared holdout of 169 trajectories. Evaluation uses the ten LIBERO-10 tasks. These trajectory counts describe policy fine-tuning.

RoboTwin evaluates the three $\pi_{0.5}$ variants, LingBot-VLA-v2, and FastWAM. The $\pi_{0.5}$ training collection covers 50 WorldArena tasks, with 500 trajectories for data20 and 1,250 for data50. Each subset has its own action-normalization statistics.

For each of the two G1 tasks, sweeping trash and collecting clothes, we evaluate four versions of the Kairos policy, labeled V1–V4 within each task.

Public policy implementations and checkpoint sources are available for $\pi_{0.5}$ through OpenPI (fine-tuning instructions; checkpoints), FastWAM (training code; checkpoints), and LingBot-VLA-v2 (fine-tuning instructions; base checkpoint; RoboTwin checkpoint). These links identify the upstream releases from which our policy variants are obtained or adapted.

A.2.3 World-model inference and action conditioning

Table 12 records each model's applied image input, sampler, temporal window, and action-conditioning interface. Inference uses the official implementation and its platform adapter. Policy prediction horizon specifies an action-sequence length, video frequency specifies timestamps in the represented trajectory, and sampling iterations specify the numerical generation procedure. None of these timeline frequencies denotes wall-clock generation throughput.

Table 12: Inference configurations used with the official implementations. R, L, and K denote RoboTwin, LIBERO, and G1. Each row groups the settings for one evaluated system; platform-specific values are marked explicitly. History/conditioning frames, future frames, and sampling iterations have distinct meanings. Guidance values follow the corresponding sampler's convention.

Model	Applied inference configuration
BWM (R)	**Image input:** Head-camera view. **Sampling:** 50 steps; classifier-free guidance (CFG) scale 1. **Temporal and action interface:** 21-frame window including 9 history frames. Forward kinematics converts joint targets into bimanual end-effector poses and gripper commands.
Ctrl-World	**Image input:** 192 × 320. **Sampling:** 25 steps; guidance 1. **Temporal and action interface:** 6 history frames and 5 output frames; 5 Hz. Actions use training-set first- and ninety-ninth-percentile normalization statistics.
Cosmos-Predict-2.5	**Image input:** L: 256 × 256; R/K: 256 × 320. **Sampling:** R: 35 steps; L/K: 25 steps; guidance 0. **Temporal and action interface:** 1 conditioning frame and 12 future frames; 5 Hz. Platform-specific 7D/14D/26D action conditioning. The consumed prediction prefix follows the actual input action length.
DreamDojo	**Image input:** 480 × 640. **Sampling:** 25 steps; guidance 0. **Temporal and action interface:** 1 conditioning frame and 12 future frames; 5 Hz. Actions map to a 384D conditioning interface. In R, each block of four actions subtracts its corresponding anchor.
DW05	**Image input:** L: letterboxed to 384 × 320. **Sampling:** R: 20 steps; L/K: 25 steps. **Temporal and action interface:** 1 conditioning frame and 8 future frames. L: 5 Hz; R/K: 7.5 Hz. L/K actions are padded to 32D.
A2World	**Image input:** 256 × 256. **Sampling:** 20 steps; guidance 0. **Temporal and action interface:** 1 conditioning frame and 20 future frames. L: 20 Hz; R/K: 5 Hz. The L adapter maps 7D policy actions to the original simulation model's 14D conditioning interface.
GE-Sim	**Image input:** 256 × 256. **Sampling:** 50 Euler steps. **Temporal and action interface:** 1 conditioning frame and 20 future frames, with 4 additional memory images. L: 20 Hz; R/K: 5 Hz. Images and robot states are jointly predicted.

Action-dimensionality changes occur within the world-model adapters; policies retain their native platform-specific action spaces.

A.2.4 Fine-tuning and official implementations

For each evaluated platform, we initialize each world model from an official checkpoint and fine-tune it for 60,000 steps on shared expert data, using the same training/test split across models. Training methods, trainable components, and optimization settings follow each model's official implementation, with platform-specific observation and action adapters. All reported results use the resulting platform-adapted checkpoints. Table 13 lists the official implementations and initialization sources.

Each world-model fine-tuning run and each closed-loop rollout worker uses eight NVIDIA A800 GPUs with 80 GB memory each. The schedule standardizes update count and hardware allocation;

batch size and computation per update remain model-specific, so training FLOPs and data exposure need not be equal.

Table 13: Official implementations and released checkpoint sources for platform-specific fine-tuning.

Model	Official implementation	Initialization source
BWM	Implementation	BWM checkpoint
Ctrl-World	Implementation	DROID-trained checkpoint
Cosmos-Predict-2.5	Implementation	2B action-conditioned checkpoint
DreamDojo	Implementation	2B pretrained checkpoint
DW05	OpenDW implementation	Base checkpoint; RoboTwin-adapted checkpoint
A2World	Implementation	Pretrained and LIBERO-adapted checkpoints
GE-Sim	V2 implementation	V2.0.1 distilled 2B checkpoint

A.3 Closed-Loop Rollout Setup

A.3.1 Reference scenes and initial conditions

RoboTwin. We use `demo_clean` with `seen` instructions. This setting retains task-specific randomization of object positions, orientations, and assets while disabling background, clutter, lighting, camera-position, and table-height perturbations. Candidate scene seeds start at 200000 and increase sequentially. Candidate scenes are checked using expert planning and task-success tests; accepted scenes are then reinitialized before executing the evaluated policy. We collect 50 accepted scenes for each task–policy pair.

The accepted seed sequences differ across RoboTwin policies. Episode indices are local to each task–policy collection. Within such a collection, all world models share the corresponding retained reference episodes. Cross-policy frequency comparisons aggregate each policy's task-level scene sample; they do not rely on episode-wise matching between policies.

LIBERO. For each task, we restore official initial-state indices 0–49 using collector seed 7 and perform ten settling steps before policy execution. These settling steps are excluded from the policy-action budget. We do not add pose or appearance perturbations to this initial-state set in the failure-type and failure-stage evaluations. Cross-policy scene pairing is established using the task definition, initial-state-file hash, and initial-state index.

G1. Each generated rollout is associated with a real execution through its task, policy version, and source UID. Initialization uses that execution's recorded initial RGB observation and 26D robot state. The source UID defines reference–model pairing. Different policy versions follow shared reset procedures but retain their own real-execution records.

A.3.2 Reference–model pairing and random seeds

Reference–model pairing fixes the task, policy checkpoint, source episode identity, initial RGB observation, initial robot state, and instruction. Within each task–policy setting, all models are scored on the same retained episode set.

Scene seeds identify candidate environments, policy seeds govern policy sampling, and generation seeds govern world-model sampling. The retained RoboTwin collection launchers specify default policy seeds of 0 for $\pi_{0.5}$ and 42 for LingBot-VLA-v2 and FastWAM. Generation follows each implementation's seed schedule; BWM, for example, uses 42 + k for generation block k. The 50 reference episodes sample different accepted scenes or initial-state indices. These are scene-level samples rather than repeated trials of one fixed scene.

A.3.3 INITIALIZATION, ACTION CONSUMPTION, AND STATE FEEDBACK

Initialization. Each world-model execution receives its source reference episode's initial RGB observation, robot state, and instruction. On RoboTwin, the BWM and DW05 implementations also use the first recorded reference action chunk for initialization. Reference-action conditioning is confined to initialization. Subsequent reference images, robot states, and actions do not condition generated execution; reference videos remain available for scoring.

Closed-loop execution. At each interaction round, the policy predicts an action sequence from the current observation and robot state, conditioned on the task instruction. Each world model advances the rollout according to its supported temporal configuration, then returns the corresponding observation and state for the next policy call. The final frame retained for the consumed action interval becomes the next observation, paired with the robot state at the same point on the action timeline.

Robot-state feedback. GE-Sim jointly predicts images and states; its predicted state corresponding to the retained image is fed to the policy. The other models propagate robot state from the consumed policy actions using their platform-specific update routines, including the corresponding coordinate and gripper conventions. These updates start from the recorded initial state and do not read subsequent reference states.

Reference execution advances through the simulator or physical robot using the same fixed policy checkpoint. After initialization, reference and generated executions obtain actions from their respective feedback streams. Their later action sequences may therefore differ even when their source initial conditions are matched.

A.3.4 ROLLOUT BUDGET

Following WorldArena (Shang et al., 2026a), we set the world-model execution budget to 1.2 times the corresponding reference duration on all three platforms. This duration is converted to each model's action and frame timeline, with rounding to its supported temporal windows. To assess possible premature truncation on RoboTwin, we sample five failed episodes that reached the original rollout limit for each task–policy–world-model configuration (20 tasks, five policies, and seven world models), yielding 3,500 episodes. We extend the rollout horizon to the task-specific maximum episode length specified by the simulator. Only 17 episodes (0.49%) succeed under the extended budget, indicating that failure outcomes in the sampled episodes are largely stable under longer evaluation horizons.

A.4 REFERENCE REPEATABILITY (REF–REF)

To assess the reproducibility of failure measurements, we independently rerun the policies in RoboTwin and LIBERO under the same evaluation settings. For failure-reason and failure-stage evaluation, we collect 50 new rollouts per task–policy setting from the same initial scenes, matching the original reference collection. We compare the two reference collections using the same annotation procedure, metrics, and aggregation rules. The resulting Ref–Ref scores provide an empirical baseline for interpreting reference–model discrepancies relative to the variation observed between independent reference executions.

For failure-boundary evaluation, Ref–Ref compares success-rate changes between two reference estimates at paired scene conditions using OA and GE (Section 3.2.3). The two estimates are obtained from independent batches of reference executions collected after the boundary pairs and their endpoint order have been fixed. Neither estimate uses the discovery rollouts, and the pairs and endpoint order remain unchanged for this comparison. Tables 6 and 19 report reference repeatability separately for position, orientation, and size changes.

B ANNOTATION PROTOCOL

This appendix defines failure labels and annotation decisions, followed by the outcome annotation prompt.

B.1 FAILURE CATEGORIES AND ANNOTATION RULES

The two interpretable failure types adapt the planning/execution distinction in REFLECT (Liu et al., 2023b) and Guardian (Pacaud et al., 2027) to trajectory annotation. Labels describe observable behavior in the context of the task instruction, without establishing internal policy causes. The examples below illustrate the definitions and are not additional measured trials.

Task Planning Failure. This category concerns task-inappropriate choices of targets, subgoals, or their ordering. It includes omitting a required step or attempting a subgoal before its prerequisites are satisfied. For example, closing a cabinet before inserting the required object, then attempting insertion through the closed door, violates a task dependency even if the closing motion succeeds.

Action Execution Failure. This category concerns failure to carry out an intended subgoal through motion or physical interaction. Examples include missing a grasp, dropping a held object during transport, and inaccurate placement. For example, the gripper approaches the intended mug but closes beside its handle and lifts without the mug. The observed interaction fails to achieve the intended grasp.

Indeterminate Failure. A confirmed failure is indeterminate when its primary type cannot be identified reliably, for example because the decisive interaction is occluded. This label is recorded as `Unjudgable`. These episodes remain in the all-episode denominator. TFE and TPC evaluate Task Planning and Action Execution.

Figure 8 illustrates observable task-planning and action-execution failures with task instructions and frame sequences.

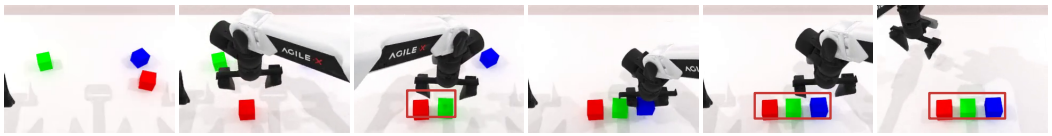
B.2 ANNOTATION DECISIONS AND OBSERVATION TIMING

Outcome assessment and verification. Final outcome labels for reference executions in RoboTwin and LIBERO follow the simulator's native task predicates. World-model rollouts and real-world G1 executions are assessed from video by two VLMs, with one expert adjudicating disagreements. To verify consistency between video-based and simulator-based outcome assessment, we compare video annotations with native task predicates on a random 10% sample of reference simulation videos, obtaining 99.8% agreement.

Type and stage assignment. For each confirmed failure, two experts label one primary type and one decisive stage; a third expert adjudicates disagreements. When several errors occur, the type follows the earliest clearly supported primary cause; the stage identifies the earliest decisive event preventing completion. Stage names are copied from the task-specific list. Timestamps and critical-moment descriptions identify repeated occurrences; stage categories follow the task-specific decomposition in Appendix A.1. If a stage cannot be identified, annotators record it as `unknown`. Reference and model executions use the same rules; successful episodes contribute zero failure counts.

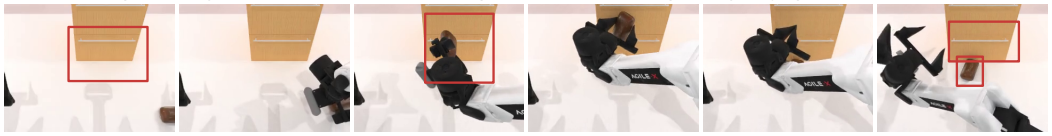
(a) Task Planning

Instruction: Form a **three-block tower**: red block bottom, green block middle, blue block top.



Evidence: The robot moves all three blocks into a **horizontal row**, which does not match the requested three-level tower.

Instruction: Open the light wood cabinet's drawer and place the smooth soap with grainy spots inside.



Evidence: The robot **skips this prerequisite** and carries and releases the soap outside the cabinet.

(b) Action Execution

Instruction: Use the right arm to pick up the brown bottle with a wide white label and set it down.



Evidence: The bottle reaches the blue pad but **tips and rolls off after release**

Instruction: Stack the three light-blue bowls one at a time.



Evidence: Placing the third bowl **knocks the stack apart**, and the robot fails to restack the scattered bowls.

Figure 8: Examples of task-planning and action-execution failures. (a) The robot arranges blocks in a row instead of the instructed tower, or attempts to place soap without first opening the drawer. (b) A released bottle tips and rolls away, or placing the third bowl disrupts the stack. Frames proceed from left to right; red boxes mark the relevant objects and interactions. These cases illustrate the two interpretable failure categories.

B.3 Outcome Annotation Prompt

The following prompt specifies video-based outcome evaluation. The placeholders `{task}` and `{instruction}` supply the task name and language instruction. The prompt requests a JSON response.

ROLE AND OBJECTIVE

You are a robotic task video evaluator. Determine whether the task was successfully completed based on the task instruction and the visible outcomes in the video.

INPUTS

Task name: `{task}`
Task instruction: `{instruction}`

EVALUATION RULES

1. Determine success by whether a visually identifiable task-completion state is achieved during the execution. The initial state alone does not establish success for a new execution. Once completion is observed, set `success=true`, even if the state changes afterward or the video continues. Do not assess whether the execution procedure was proper.
2. A blurry but reasonably identifiable completion state is sufficient for `success=true`. If no completion state is observed and the visible evidence supports that the task remains incomplete at the end of the video, set `success=false`.
3. For generated videos, ignore visual artifacts when they do not prevent outcome assessment. Focus on whether the task goal was achieved. Lower confidence when uncertain.
4. Rely only on the task instruction and visible evidence in the video. Do not infer the outcome from success/failure folder names or model names.
5. If the available visual evidence is insufficient to determine whether the task was completed, set `success="unjudgeable"`. Blur, a limited camera view, or an object being out of view does not by itself imply failure; use "unjudgeable" when these limitations prevent a reliable outcome judgment.
6. `success_timestamp_sec` is the approximate time in seconds when the task completion state first appears. Set it to null when `success=false` or `success=unjudgeable`.
7. Include only visually observable evidence in evidence. Do not classify the failure type or failure stage in this evaluation.

In the JSON response, success must be true, false, or the string "unjudgeable".

OUTPUT FORMAT

Return only valid JSON, without Markdown:

```
{
  "success": true,
  "success_timestamp_sec": 0.0,
  "reason": "<rationale and uncertainty>",
  "evidence": [
    "<visible evidence>"
  ],
  "confidence": 0.0
}
```

C Boundary Construction and Interventions

We construct 60 boundary pairs in each of RoboTwin and LIBERO, with 20 per factor. Starting from successful scenes, we intervene on one object's position, orientation, or size while holding the instruction, policy weights, and remaining scene settings fixed. RoboTwin pairs use LingBot-VLA-v2, $\pi_{0.5}$, and FastWAM; LIBERO uses the full-data, data-50, and data-20 $\pi_{0.5}$ variants.

C.1 Intervention Ranges

Table 14 distinguishes RoboTwin's preset search ranges from its selected endpoints. Position and rotation are offsets from the initial scene; size multipliers are relative to the object's original linear dimensions. On RoboTwin, cube rotation paths are restricted to $\pm 40^\circ$, and the middle object in size-ranking tasks permits only multipliers 0.9, 1.0, and 1.1; these special paths contribute no final pairs. For endpoint multipliers s_1 and s_2, the relative size change is $100|s_2/s_1 - 1|\%$.

LIBERO uses the same preset search ranges, with 4 cm position steps and 0.2 scale-factor steps. Its yaw step is 10°, compared with RoboTwin's 20°. Both platforms follow the discovery and evaluation procedure below.

Table 14: RoboTwin boundary search ranges, grid steps, and selected endpoint ranges. Position and yaw are offsets from the initial scene; size is a linear scale multiplier. LIBERO shares the search ranges and uses a 10° yaw step.

Intervention	Search range	Pair step	Selected endpoint range
Planar position (x or y)	-40 to 40 cm	4 cm	-36 to 28 cm
Yaw about the vertical axis	-120° to 120°	20°	-120° to 120°
Linear size multiplier	0.4–2.0	0.2	0.4–1.8

C.2 Boundary Discovery and Independent Evaluation

Both platforms separate boundary discovery from final reference estimation using the following procedure.

(1) Discover candidate boundaries. We estimate success at adjacent conditions on each preset path using ten complete reference rollouts per endpoint. A pair qualifies when the success rates differ by at least 30 percentage points, equivalent to a difference of three successes out of ten. Both increases and decreases along the scan direction qualify. Physical pairs are deduplicated, and 20 pairs are selected for each intervention factor in each platform.

(2) Freeze the evaluation set. Before independent reference re-estimation, we fix all 60 pairs per platform, including their tasks, policies, base scenes, and endpoint conditions. Endpoints are ordered from higher to lower success according to the discovery estimates. The set and endpoint order remain fixed throughout evaluation: no pair is removed or reordered based on the re-estimated success rates, even if its difference falls below 30 percentage points or reverses direction.

(3) Independently re-estimate reference success. At every frozen endpoint in both platforms, we collect ten new reference executions using random seeds disjoint from those used for discovery. Success follows the simulator's task predicate. All complete evaluation executions enter the estimate; exploratory probes, invalid scenes, and infrastructure errors are not counted as task failures. Only these ten new executions determine the reference success rate used for final evaluation; discovery executions are not pooled into this estimate.

(4) Evaluate world models. Each world model runs the same policy ten times at each frozen endpoint, from the corresponding initialization. OA and GE (Section 3.2.3) compare model success-rate changes with the independently re-estimated reference changes, using the frozen endpoint order. The reported fraction of underestimated reference changes also uses these independent estimates.

C.3 Scene Examples

Figure 9 illustrates RoboTwin's position, orientation, and size interventions. Each row changes one scene factor while keeping the task and policy fixed.

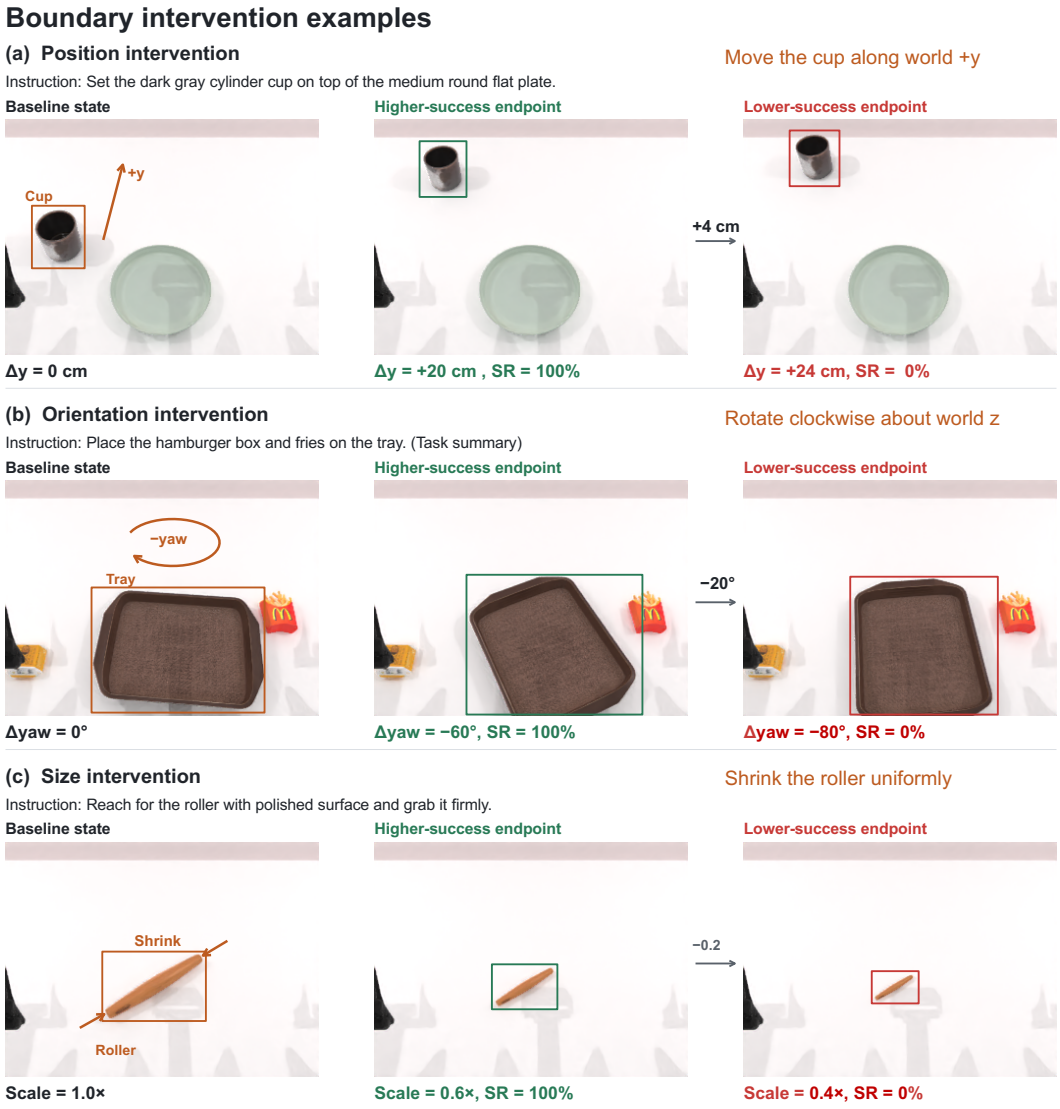


Figure 9: Scene interventions for boundary construction. SR denotes success rate; Δy and $\Delta\mathrm{yaw}$ are offsets from the base scene, with scale factors relative to the original object size. The conditions differ by (a) cup translation along world $+y$, from $+20$ to $+24\,\mathrm{cm}$ (success: 100% to 0%); (b) tray rotation about world z, from -60° to -80° (100% to 0%); or (c) uniform roller scaling, from $0.6\times$ to $0.4\times$ (100% to 0%).

D Additional Experimental Results

D.1 RoboTwin Failure-Type Policy Correlation

Table 15 reports planning, execution, and aggregate TPC on RoboTwin.

Table 15: Type-specific Policy Correlation (TPC) on RoboTwin (↑). Type columns average defined within-task Pearson correlations across five policies. Overall Mean averages defined type correlations within tasks, then tasks equally. Bold marks the best world-model values.

Evaluator	Task Planning	Action Execution	Overall Mean
Ref–Ref	0.796	0.831	0.814
A2World	0.353	−0.086	0.134
BWM	−0.121	0.143	0.011
Cosmos	0.008	0.038	0.023
Ctrl	0.177	0.204	0.191
DW05	**0.456**	**0.348**	**0.402**
DreamDojo	0.102	−0.306	−0.102
GE-Sim	0.336	0.142	0.239

D.2 RoboTwin Failure-Stage Profiles Across Policies

Figure 10 compares stage-specific failure frequencies across all five policies on `blocks_ranking_size`. For every policy, reference failures peak at Grasp small (S2), while DreamDojo's generated failures peak at Place (S4).

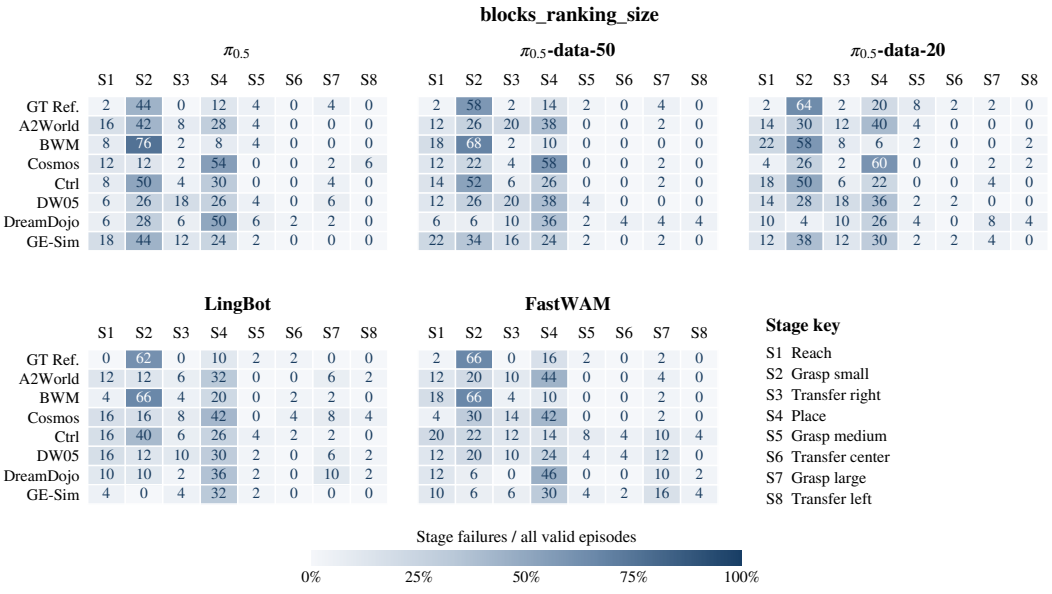


Figure 10: Stage-specific failure frequencies for `blocks_ranking_size` across five policies. Each panel shows reference execution and seven world models; cells report the percentage of all valid episodes with a failure at each stage. The stage key and color scale are shared.

D.3 RoboTwin Failure-Boundary Responses

Figure 11 shows success-rate changes for individual boundary pairs and the shares of preserved, flat, and reversed responses for each intervention factor.

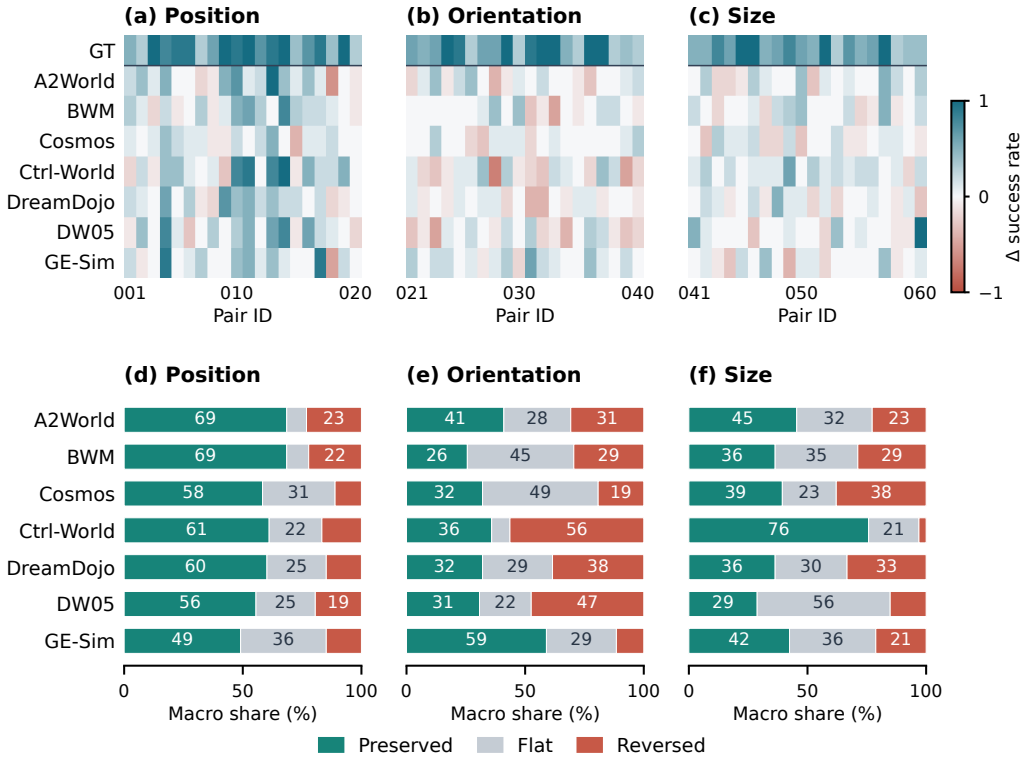


Figure 11: Responses across 60 frozen boundary pairs. **(a–c)** Reference and model success-rate changes, labeled Δ^{GT} and Δ^{WM}, respectively. Each column uses the endpoint order fixed during discovery; reference changes come from independent re-estimation. **(d–f)** Setting-weighted shares of preserved, flat, and reversed responses. Model changes within ± 2 percentage points are flat; nonflat changes are preserved or reversed according to agreement with the independently estimated reference direction. The 95.2% underestimation rate is descriptive: the 420 model–pair comparisons across seven models share boundary pairs and some endpoints and are not independent.

D.4 LIBERO Failure-Type Understanding

We evaluate six world models on LIBERO-10 using $\pi_{0.5}$ and its data-50 and data-20 variants. Table 16 reports TFE.

Table 16: Type Frequency Error (TFE) on LIBERO (↓). Policy columns average tasks; Mean weights the three policies equally. Bold marks the best world-model values.

Evaluator	$\pi_{0.5}$	$\pi_{0.5}$-data-50	$\pi_{0.5}$-data-20	Mean
Ref–Ref	0.032	0.056	0.048	0.045
A2World	0.300	0.264	**0.248**	**0.271**
Cosmos	0.322	0.288	0.254	0.288
Ctrl	**0.290**	**0.257**	0.282	0.276
DW05	0.352	0.293	0.284	0.310
DreamDojo	0.421	0.384	0.338	0.381
GE-Sim	0.352	0.324	0.351	0.342

Type Frequency Error (TFE). Generated rollouts substantially misestimate the prevalence of policy failure types. Mean TFE ranges from 0.271 to 0.381, compared with 0.045 for Ref–Ref. Ctrl has the lowest TFE for the full-data and data-50 policies, whereas A2World is most accurate for data-20: evaluator accuracy varies even within one policy family.

Table 17: LIBERO TPC across three policies (↑). Bold marks the best world-model values.

Evaluator	TPC
Ref–Ref	0.780
A2World	−0.062
Cosmos	**0.242**
Ctrl	−0.189
DW05	0.043
DreamDojo	0.108
GE-Sim	0.029

Type-specific Policy Correlation (TPC). World models poorly preserve cross-policy differences in failure-type frequencies. TPC ranges from −0.189 to 0.242, below Ref–Ref's 0.780 (Table 17).

D.5 LIBERO FAILURE-STAGE UNDERSTANDING

Table 18 reports SFE and SPC for the same three policy variants. Figure 12 illustrates stage-specific failure frequencies on `task_00`, which requires placing both the alphabet soup and tomato sauce in a basket.

Table 18: Stage Frequency Error (SFE, ↓) and Stage-specific Policy Correlation (SPC, ↑) on LIBERO. SFE policy columns average tasks; Mean averages policies within tasks, then tasks. Frequencies use all valid paired episodes. SPC averages defined within-task Pearson correlations over stages, then tasks. Bold marks the best world-model values.

	SFE (↓)				SPC (↑)
Evaluator	$\pi_{0.5}$	$\pi_{0.5}$-data-50	$\pi_{0.5}$-data-20	Mean	
Ref–Ref	0.064	0.124	0.140	0.109	0.680
A2World	0.808	0.796	**0.768**	0.791	−0.271
Cosmos	0.836	0.832	0.824	0.831	−0.173
Ctrl	**0.692**	**0.682**	0.844	**0.739**	−0.021
DW05	0.932	0.862	0.928	0.907	−0.235
DreamDojo	0.998	0.988	0.988	0.991	0.052
GE-Sim	0.856	0.844	0.938	0.879	**0.228**

Stage Frequency Error and Stage-specific Policy Correlation. Low frequency error does not ensure faithful cross-policy comparisons of stage-failure frequencies. Ctrl has the lowest mean SFE (0.739), but its SPC is −0.021. Conversely, GE-Sim has the highest SPC (0.228) despite a larger SFE (0.879). Ref–Ref achieves SFE 0.109 and SPC 0.680, while four of six models have negative SPC. SFE and SPC capture complementary aspects of the full-task stage-failure profile: frequency accuracy and cross-policy variation.



Figure 12: Stage-specific failure frequencies for LIBERO `task_00` across three policies. Each panel compares reference execution with six world models; cells give the percentage of all valid episodes with a failure at each stage. The stage key and color scale are shared. The data-100 policy is the full-data $\pi_{0.5}$ used in the text and tables.

Case study: exaggerated early-stage failures. Correctly locating the dominant failure stage does not ensure accurate failure-frequency estimates. In `task_00`, reference failures at Grasp soup (S1) occur in 18%, 36%, and 26% of episodes for the three policies, whereas DreamDojo predicts 100%, 100%, and 98%. Although both identify S1 as the peak, the generated frequencies greatly overstate the prevalence of early grasp failures during full-task execution.

D.6 LIBERO Failure-Boundary Understanding

Table 19 reports OA and GE for six world models and the Ref–Ref baseline on 60 LIBERO boundary pairs.

Table 19: Failure-boundary results on LIBERO (OA ↑, GE ↓). Ref–Ref reports reference repeatability. Bold marks the best world-model value in each column, including ties.

Evaluator	Position		Orientation		Size	
	OA ↑	GE ↓	OA ↑	GE ↓	OA ↑	GE ↓
Ref–Ref	0.950	0.035	0.850	0.080	0.900	0.065
A2World	0.067	**0.517**	0.150	0.584	0.000	0.584
Cosmos-Predict-2.5	0.067	0.522	0.075	0.534	0.000	0.556
Ctrl-World	0.000	0.528	**0.175**	**0.514**	0.077	0.543
DreamDojo	0.033	0.522	0.000	0.543	0.000	0.530
DW05	0.000	0.528	0.000	0.555	0.000	0.549
GE-Sim	**0.089**	**0.517**	0.150	0.545	**0.154**	**0.504**

Failure-boundary evaluation. **(1) World models rarely identify the correct direction of policy sensitivity on LIBERO.** OA ranges from 0 to 0.175 across all model–factor combinations, limiting their ability to identify which nearby condition is less reliable. Ref–Ref instead attains OA of 0.85–0.95 across the three factors. **(2) World models do not reliably quantify how scene changes affect success rates.** Even the lowest world-model GE for each factor remains 0.504–0.517, corresponding to errors of 50.4–51.7 percentage points, compared with Ref–Ref GE of 0.035–0.080.

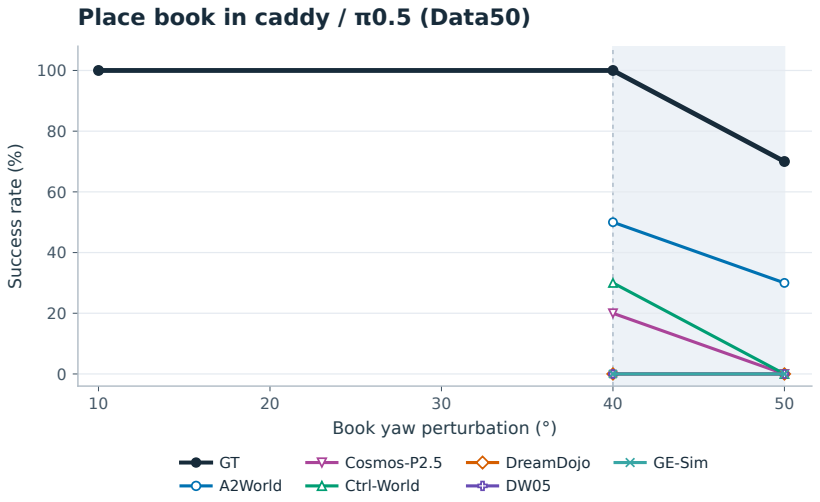


Figure 13: Orientation case for $\pi_{0.5}$-data-50 on the LIBERO task of placing a book in a caddy. The reference curve shows sampled book yaw perturbations; world-model curves connect the highlighted $40°$ and $50°$ endpoints. Cosmos-P2.5 denotes Cosmos-Predict-2.5.

Case study: a missed orientation boundary. **Uniformly pessimistic predictions can hide differences between nearby conditions.** As book yaw increases from $40°$ to $50°$, reference success falls from 100% to 70%. DreamDojo, DW05, and GE-Sim instead predict 0% at both endpoints (Figure 13). These evaluators would label both conditions as uniformly unreliable, obscuring the reference policy's loss of reliability under the orientation change.

D.7 Real-World Results on G1

We evaluate six world models on sweeping trash and collecting clothes, using four task-specific policy versions, V1–V4, for each task. Version labels do not imply a performance ordering. Table 20 reports TFE and SFE. Table 22 gives reference success rates, with V3 performing best in both tasks.

Table 20: Failure-diagnostic errors for real-world tasks on G1 (↓). TFE averages two failure-type errors; SFE sums four stage errors. Task scores average four policy versions; Mean weights the two tasks equally. Bold marks column minima.

	Failure type: TFE			**Failure stage: SFE**		
World model	Sweep	Clothes	Mean	Sweep	Clothes	Mean
Ctrl	0.0971	0.1195	0.1083	0.8284	0.4042	0.6163
DreamDojo	0.1528	0.1715	0.1621	0.8340	0.5800	0.7070
Cosmos	0.1796	0.2929	0.2362	0.7143	0.9544	0.8343
DW05	0.1429	0.2171	0.1800	0.6436	0.7269	0.6852
A2World	**0.0603**	0.1258	0.0930	**0.2759**	0.3975	**0.3367**
GE-Sim	0.0838	**0.0992**	**0.0915**	0.5620	**0.3829**	0.4725

Type and Stage Frequency Errors. The most accurate evaluator depends on the task and the evaluation dimension. A2World has the lowest sweeping TFE and SFE (0.0603 and 0.2759), while GE-Sim has the lowest clothes-collection errors (0.0992 and 0.3829). Across tasks, GE-Sim has the lowest mean TFE (0.0915), whereas A2World has the lowest mean SFE (0.3367).

Table 21: Task-specific Pearson agreement across four policy versions for real-world tasks on G1. TPC and SPC average defined type and stage correlations, respectively. Bold marks column maxima.

	Sweeping trash		Collecting clothes	
Model	TPC (↑)	SPC (↑)	TPC (↑)	SPC (↑)
Ctrl	0.5016	-0.1270	**0.6812**	0.3463
DreamDojo	-0.4996	0.1648	0.3714	**0.4824**
Cosmos	-0.6619	-0.2548	0.1005	0.2258
DW05	0.1046	-0.0251	0.0569	0.1479
A2World	**0.5118**	**0.4667**	-0.2089	0.0545
GE-Sim	0.4406	-0.2479	0.3014	0.2336

Table 22: Reference success rates (%) for real-world tasks on G1, computed over executions with judged outcomes. Each task has four policy versions; bold marks the highest rate within each task.

Policy	Sweeping trash	Collecting clothes
V1	30.61	28.00
V2	50.00	59.18
V3	**65.31**	**78.00**
V4	28.00	56.25

Type- and Stage-specific Policy Correlation. (1) Evaluator fidelity does not transfer uniformly between real-world tasks. A2World has the highest sweeping TPC and SPC (0.5118 and 0.4667), but drops to -0.2089 and 0.0545 for clothes collection (Table 21). **(2) Preserving type differences does not ensure faithful comparisons of stage-failure frequencies.** GE-Sim has a sweeping TPC of 0.4406 but an SPC of -0.2479. For clothes collection, Ctrl has the highest TPC (0.6812), whereas DreamDojo has the highest SPC (0.4824). Evaluator suitability therefore depends on both the task and the diagnostic dimension.

E Qualitative Reference and World-Model Comparisons

The following examples illustrate preserved failures, failure-type mismatches, and failure-stage mismatches through reference and generated frame sequences. The displayed annotations identify the behaviors and stages used in each comparison.

E.1 Simulation

Figure 14 compares reference executions with Ctrl-World, BWM and A2World on tool use, blocks arrangement, and bowl stacking.

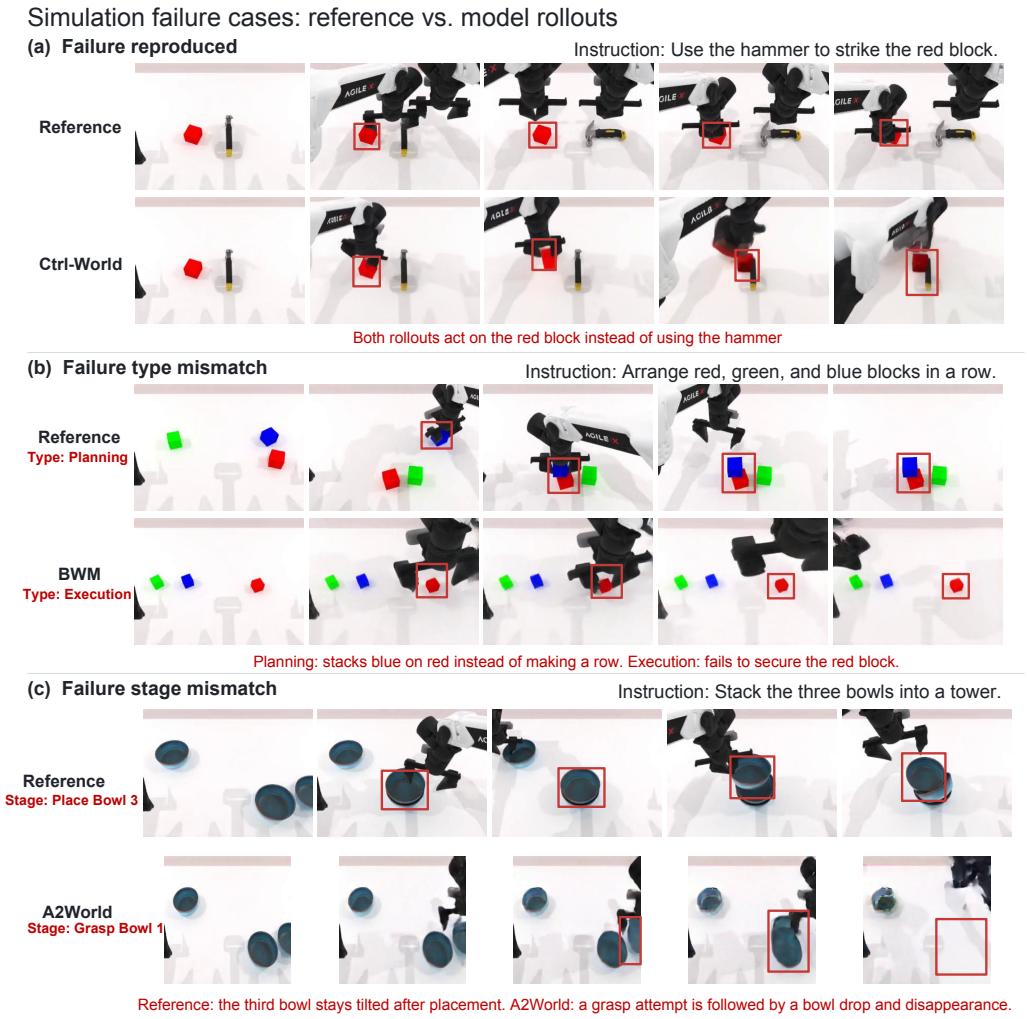


Figure 14: Qualitative failure comparisons between reference and world-model rollouts in simulation. (a) Reference and Ctrl-World both manipulate the red block directly instead of using the hammer. (b) The blocks-ranking-color example shows different annotated failure types: the reference arranges the blue block on top of the red one in instead of in a row, while the model fails to acquire the red block. (c) The reference fails during placement of the third bowl, while A2World fails during acquisition of the first. Frames proceed from left to right, with red boxes marking the displayed evidence.

E.2 Real-World Tasks on G1

Figure 15 compares reference executions with A2World and DreamDojo on sweeping trash and collecting clothes.

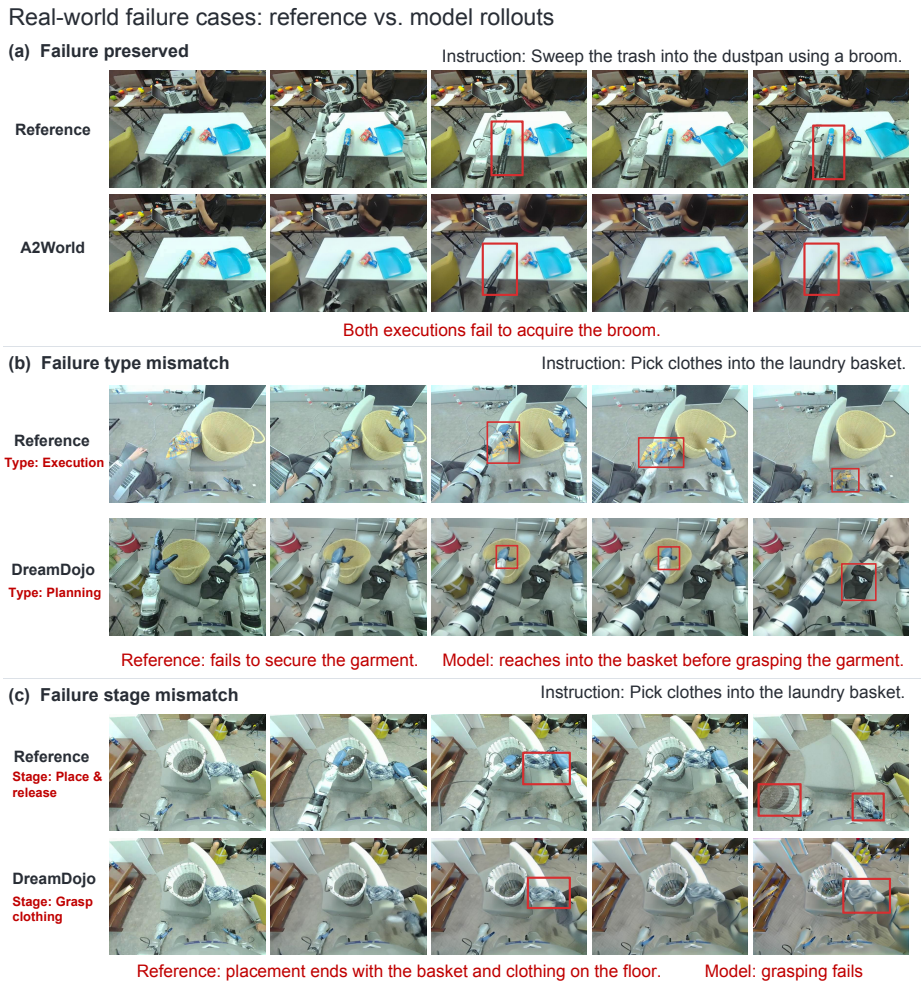


Figure 15: Reference and world-model rollouts for real-world tasks on G1, illustrating (a) preserved failure, (b) a failure-type mismatch, and (c) a failure-stage mismatch. Frames proceed from left to right; instructions, labels, and visual evidence appear in each panel.

Reproducibility Statement

Task and experimental settings, annotation, and boundary construction are specified in Appendices A–C. Section 3.2 defines the metrics.

AI Use Statement

AI assistance was used to review related work, reconcile experimental outputs, and prepare the manuscript and figures. Doubao-Seed-2-1-Pro and ChatGPT 6 were used for rollout outcome annotation, with expert adjudication of disagreements and random verification of agreed labels.