# ACTIONTRACE: LEARNING ACTION CONSEQUENCES IN ROBOT WORLD MODELS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Robot world models offer a way to evaluate policies through imagined rollouts, provided that their predictions reflect the consequences of the supplied actions. However, from the same initial scene, a model may generate similar, visually plausible futures for actions with different physical outcomes, obscuring differences between policies. To preserve these distinctions, we introduce ActionTrace, a robot world model that learns action consequences through spatial and temporal conditioning. It projects joint trajectories into the camera view to guide predicted robot motion and carries an action-updated consequence state across video segments, allowing earlier actions to inform later predictions. We reinforce this action dependence during training by favoring executed controls over alternatives from the same simulator reset for each observed future. Under paired action interventions, ActionTrace predicts object outcomes with higher overall accuracy across recursive rollouts. Across four newly trained policies on 50 manipulation tasks, ActionTrace recovers their ordering with a Pearson correlation of 0.9805 between predicted and simulator success rates. On the WorldArena benchmark, it also recovers the ordering of five released policies, reaching a Pearson correlation of 0.9920. The model achieves these policy-evaluation gains while retaining comparable long-horizon video fidelity. Adapting its components to Ctrl-World extends these policy-evaluation improvements to real-world outcomes across seven tasks and three policies.

## 1 INTRODUCTION

Action-conditioned world models predict future observations from the current scene and a supplied sequence of robot actions. They can serve as learned environments for policy evaluation: a policy acts on generated observations, and its performance is estimated from the resulting rollouts (Guo et al., 2026b; Li et al., 2025b; Quevedo et al., 2026; Shang et al., 2026). Evaluating policies this way requires predictions to reflect the consequences of both successful and unsuccessful actions.

The initial scene and task instruction can already suggest much of a familiar manipulation sequence. A video model may therefore produce a plausible continuation without accurately following the supplied actions. When actions with distinct physical outcomes yield overly similar predicted futures, we call this *action-consequence collapse* (Figure 1).

To distinguish these futures, a world model needs to follow how actions change the scene. This begins with connecting joint-space commands to robot motion in the camera view. The resulting interactions then determine what happens next: the same lifting command should carry an object after a successful grasp but leave it behind after a missed grasp. Predictions across video segments therefore need to retain the consequences of earlier interactions.

We introduce ActionTrace, a robot world model that incorporates these spatial and temporal dependencies into video prediction (Figure 2). We train the model with same-reset branch supervision, contrasting policy executions from a shared simulator reset to strengthen the connection between actions and predicted outcomes.



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.

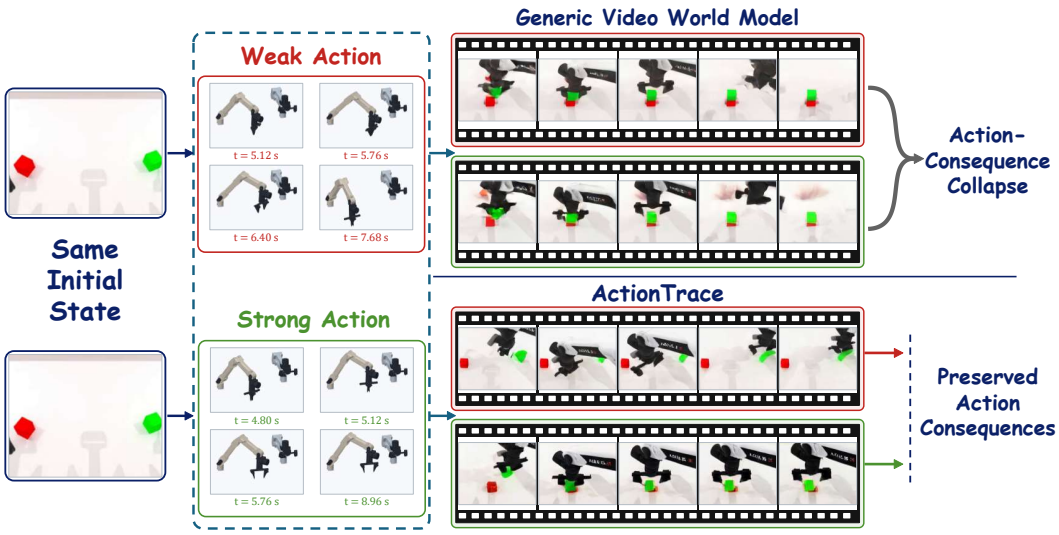


Figure 1: Illustration of action-consequence collapse. Given the same initial state, a world model can generate similar, apparently successful futures for actions with different physical outcomes. ActionTrace preserves action-dependent differences in the predicted futures.

Our contributions are threefold:

- We propose ActionTrace, a robot world model that learns action consequences through spatial and temporal conditioning. Projected joint trajectories guide motion in the camera view, while an action-updated consequence state carries information across video segments to condition later predictions.
- We introduce same-reset branch supervision using policy trajectories collected from shared initial states. By favoring executed actions over alternatives for each observed future, it strengthens action-dependent prediction and complements visual and terminal-outcome supervision of the recurrent state.
- We show that ActionTrace predicts paired object-interaction outcomes more accurately and improves policy ranking while maintaining comparable long-horizon video fidelity. Adapting its components to Ctrl-World improves agreement with real-world outcomes.

## 2 Related Work

**Robot world models and policy evaluation.** Action-conditioned video models predict how environments evolve under supplied controls, from early applications in Atari games and robotic visual planning (Oh et al., 2015; Finn & Levine, 2017) to interactive simulation in UniSim (Yang et al., 2024). By jointly modeling observations and actions, PAD (Guo et al., 2024), UWM (Zhu et al., 2025a), and recent world-action models (Guo et al., 2026a; Yuan et al., 2026; Pan et al., 2026) extend this predictive modeling to policy learning. Predicting the outcomes of policy actions also provides a basis for policy evaluation. SIMPLER studies how policy performance in physics-based simulation corresponds to real-world performance (Li et al., 2025a), while video world models assess policies through generated rollouts (Zhu et al., 2025b; Li et al., 2025b; Quevedo et al., 2026; Guo et al., 2026b; Li et al., 2026). WorldArena benchmarks video world models in this role (Shang et al., 2026). For these evaluations to distinguish policies, the generated outcomes must reflect differences in the consequences of their actions.

**Action conditioning and rollout consistency.** Action conditioning connects robot controls to generated motion: frame-aligned control features (Zhu et al., 2025b) provide temporal cues, while rendered kinematic skeletons (Wang et al., 2025; Wu & Gao, 2026) express the commanded motion in the camera view. As predictions extend across video segments, the model also needs context from earlier interactions. Prior work retains this context through historical observations or memory (Guo

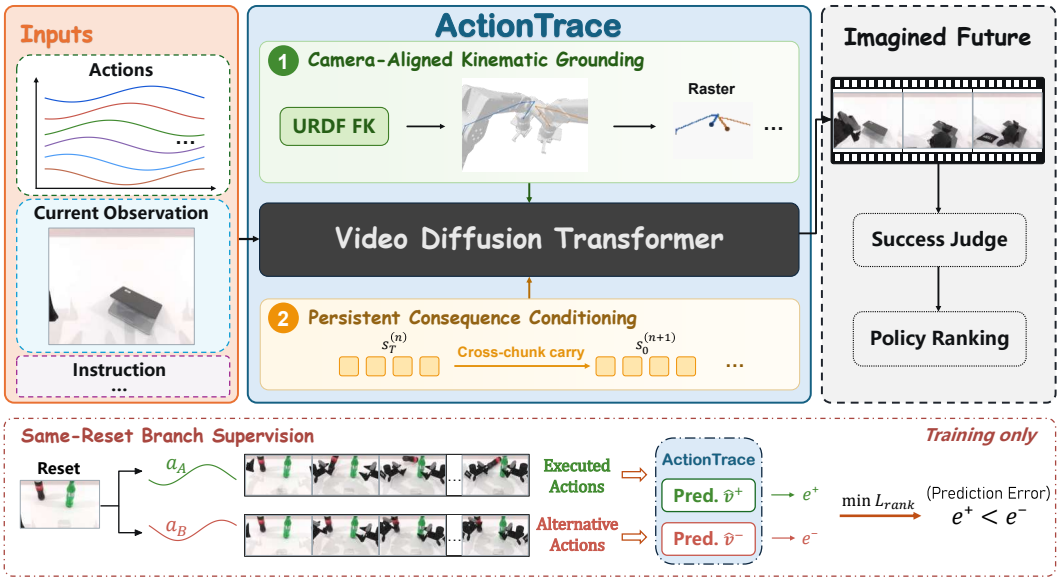


Figure 2: Overview of ActionTrace. The model grounds robot motion in the camera view and carries a consequence state across video chunks to predict the effects of supplied actions. During training, it compares how well executed and alternative controls from the same reset explain the same recorded future, with a ranking loss favoring the executed controls. At evaluation, we rank policies by their success rates in its generated rollouts.

et al., 2026b; Li et al., 2026; Zheng et al., 2026), while training with generated context helps the model continue from its own predictions (Huang et al., 2025; Jeon et al., 2026). ActionTrace uses spatial and temporal conditioning to preserve action-dependent differences in robot motion and object outcomes across video segments. We reinforce this action dependence through same-reset branch supervision.

# 3 Method

## 3.1 Problem Formulation

Let $o_t$ denote the current RGB observation, $q_t \in \mathbb{R}^{14}$ the current dual-arm joint command, $\ell$ the task instruction, and $a_{t:t+H-1}$ an $H$-step action sequence supplied by policy $\pi$. We represent each action as the difference between the policy's target joint configuration and the corresponding absolute joint command. Conditioned on these controls and an initial consequence state $s_0$, our video model predicts

$$p_\theta(o_{t+1:t+H} \mid o_t, \ell, q_t, a_{t:t+H-1}, s_0)\,. \tag{1}$$

We predict the supplied sequence in consecutive rollout chunks, using each chunk's final frame to condition the next. At replanning steps, the policy supplies a new sequence from the latest generated observation.

**Architecture.** ActionTrace uses X-WAM's 5B-parameter video Diffusion Transformer (DiT) (Guo et al., 2026a), built on Wan pretraining (Team Wan et al., 2025) (Figure 3). For each $H = 16$-step action sequence, it predicts sixteen future frames from one observed frame. The video VAE represents this chunk with one observed and four future latent positions. At flow time $\tau$, the video input $z_\tau$ combines clean observed latents with noisy future latents. Conditioned on the task instruction and numeric tokens encoding joint commands and actions, the DiT denoises the future latents, which are decoded into RGB frames.

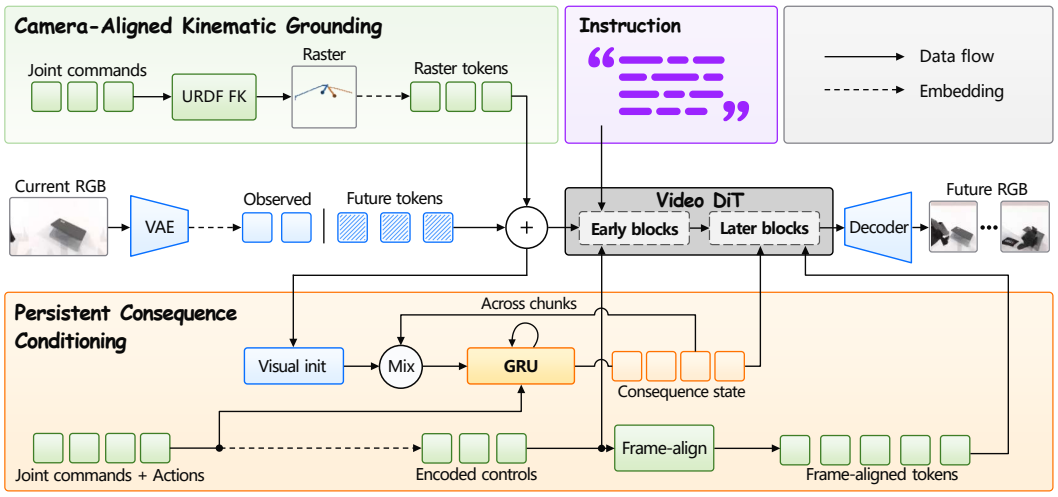


Figure 3: ActionTrace architecture. Kinematic grounding adds projected motion cues at the video input, while consequence conditioning supplies a recurrent state and frame-aligned controls to later DiT blocks. The return connection carries the state across chunks.

### 3.2 Camera-Aligned Kinematic Grounding

Starting from the current joint command, we form a candidate joint trajectory from the policy's target configurations. Robot kinematics and camera calibration map this trajectory into the camera view. Let $\mathcal{U}$ be the URDF kinematic tree and $F_j(q;\mathcal{U}) \in \mathbb{R}^3$ the position of link point $j$ under commanded joint configuration $q$. For camera intrinsics $K$ and extrinsics $[R \mid t]$, we project each link point as

$$\widetilde{u}_{k,j} = K[R \mid t] \begin{bmatrix} F_j(q_k;\mathcal{U}) \\ 1 \end{bmatrix}, \qquad u_{k,j} = \left( \frac{\widetilde{u}^x_{k,j}}{\widetilde{u}^z_{k,j}}, \frac{\widetilde{u}^y_{k,j}}{\widetilde{u}^z_{k,j}} \right). \tag{2}$$

We render the projected sequence as a four-channel raster $r$. Two channels contain depth-coded skeletons for the left and right arms; the other two contain end-effector markers whose intensity represents each gripper's openness. The raster follows the RGB image crop and is sampled and resized to match the video's latent temporal and spatial grid.

The raster encoder $\phi_r$ uses a three-dimensional convolution to map the four raster channels to the video hidden width with the same temporal and spatial patch lattice as the video embedding $\phi_v$. The input tokens are

$$x_0 = \phi_v(z_\tau) + \lambda_r \phi_r(r) + e_{\text{view}}, \tag{3}$$

where $\lambda_r$ weights the raster features and $e_{\text{view}}$ is the view embedding.

### 3.3 Persistent Consequence Conditioning

We maintain a recurrent consequence state to carry information from earlier interactions into later predictions, alongside frame-aligned controls for the current action interval. We initialize this state from the current scene with learned queries $Q$ attending to the observed-frame tokens $v_0$ after raster addition (Eq. 3; Visual init in Figure 3):

$$s_0 = Q + \text{Attn}(Q, v_0, v_0). \tag{4}$$

We then advance the state along the candidate actions, using one update per four actions to match the VAE's latent temporal intervals. At latent position $k$, let $q_k$ be the absolute joint command and $\bar{a}_k$ the mean action over the following interval. A GRU updates the state from these controls, and we project each state to the video hidden width as $c_k$:

$$h_k = \text{MLP}([\bar{a}_k, q_{k+1}, q_{k+1} - q_k]), \tag{5}$$

$$s_{k+1} = \text{GRUCell}(h_k + Q, s_k), \tag{6}$$

$$c_k = W_c\, \text{LN}(s_k). \tag{7}$$

To carry this information across video segments, we mix the terminal state $s_T^{(n)}$ of chunk $n$ with the visual initialization $s_{\text{vis}}^{(n+1)}$ from the next chunk's boundary frame:

$$s_0^{(n+1)} = \alpha s_T^{(n)} + (1-\alpha) s_{\text{vis}}^{(n+1)}, \tag{8}$$

where $T$ denotes the final latent-time index of each chunk and $\alpha$ weights the carried state.

We use the resulting state sequence to guide video prediction at corresponding latent times. In later block $l$, the projected state $c_k$ conditions spatial video tokens $x_k^l$ through cross-attention and a direct residual:

$$\widetilde{x}_k^l = x_k^l + g_l \,\mathrm{Attn}(\mathrm{LN}(x_k^l), c_k, c_k) + \gamma W_r \bar{c}_k, \tag{9}$$

where $g_l$ is a learned gate, $\gamma$ weights the direct residual, and $\bar{c}_k$ is the mean over the projected state tokens $c_k$. The residual is broadcast over spatial positions.

Alongside the recurrent state, we provide direct control features for the current action interval. These features encode joint commands and actions on the VAE's latent time grid and are added at matching temporal positions in the later blocks (Figure 3, frame-alignment pathway; Appendix A.2).

We train the state to represent visual consequences and task outcomes. For the executed controls, we align $\bar{c}_k$ with $\bar{v}_k$, the spatial mean of clean-video features from $\phi_v$ at future latent time $k$:

$$\mathcal{L}_{\text{lat}} = \frac{1}{T} \sum_{k=1}^{T} \left[1 - \cos(\bar{c}_k, \mathrm{sg}(\bar{v}_k))\right], \tag{10}$$

where $\cos$ denotes cosine similarity and $\mathrm{sg}$ stops gradients through the visual target. To encode task outcomes, a training-only head predicts success probability $p = \sigma(f_{\text{out}}(\bar{s}_T))$, where $\bar{s}_T$ is the mean over the end-of-chunk state tokens $s_T$ and $\sigma$ is the sigmoid function. For clips reaching the trajectory end, we apply binary cross-entropy with the trajectory's success label $y$:

$$\mathcal{L}_{\text{out}} = -y \log p - (1-y) \log(1-p). \tag{11}$$

### 3.4 Training objectives

We train kinematic grounding and consequence conditioning separately before joint training, using video prediction and same-reset branch supervision for both components. For video prediction, we use flow matching on recorded action–video trajectories (Lipman et al., 2023). Given clean future latents $z_{0,\text{fut}}$ and Gaussian noise $\epsilon$, we form $z_{\tau,\text{fut}} = (1-\tau) z_{0,\text{fut}} + \tau\epsilon$. The model predicts the velocity $\epsilon - z_{0,\text{fut}}$, and $\mathcal{L}_{\text{flow}}$ measures its weighted mean squared error over future positions.

**Same-reset branch supervision.** To distinguish actions with different consequences, we compare how well executed and alternative controls explain the same recorded future (Figure 2, training inset). For trajectories $A$ and $B$ collected from the same simulator reset, the factual pass uses $A$'s controls and the paired pass uses $B$'s controls to predict $A$'s recorded future. Let $e^+$ and $e^-$ denote their respective flow-prediction errors, computed against the same velocity target with the same future-position weights. We favor the executed controls using the margin-based loss

$$\mathcal{L}_{\text{rank}} = \left[m + e^+ - e^-\right]_+, \tag{12}$$

where $m$ is the ranking margin. During joint training, this loss uses reset-start clips with distinct controls and visibly different recorded futures. Algorithm 1 summarizes this comparison.

**Joint training.** ActionTrace inherits the consequence-conditioning model and the kinematic-grounding variant's raster encoder. We add direct state residuals and cross-chunk carry, then jointly optimize video prediction, branch ranking, and the state-supervision objectives in Eqs. 10 and 11:

$$\mathcal{L} = \mathcal{L}_{\text{flow}} + \lambda_{\text{lat}} \mathcal{L}_{\text{lat}} + \lambda_{\text{out}} \mathcal{L}_{\text{out}} + \lambda_{\text{rank}} \mathcal{L}_{\text{rank}}. \tag{13}$$

Appendix A.3 details the stage-specific pairing rules and the full training procedure, including how state-supervision losses are assigned to factual and paired passes.

**Algorithm 1 Same-reset branch supervision**

**Given:** World-model flow predictor $f_\theta$; reset-start clips from $A, B$ meeting the branch-selection criteria; task instruction $\ell$.

1: Extract actions, joint commands, and rasters: $(a^A, q^A, r^A)$ and $(a^B, q^B, r^B)$.
2: Encode the shared initial image as clean observed latents and $A$'s future video as $z_{0,\mathrm{fut}}$.
3: Sample $\epsilon, \tau$; combine the observed latents with $(1-\tau)z_{0,\mathrm{fut}} + \tau\epsilon$ to form $z_\tau$.
4: $\hat{v}^+ \leftarrow f_\theta(z_\tau, \tau, \ell, q^A, a^A, r^A)$.
5: $\hat{v}^- \leftarrow f_\theta(z_\tau, \tau, \ell, q^B, a^B, r^B)$.
6: Set the shared velocity target $u \leftarrow \epsilon - z_{0,\mathrm{fut}}$.
7: Compute errors $e^+, e^-$ against $u$ with the same future-position weights.
8: **return** $\mathcal{L}_{\mathrm{rank}} = [m + e^+ - e^-]_+$.

# 4 EXPERIMENTS

We first examine whether ActionTrace better preserves action-dependent object outcomes and whether this improvement translates into better closed-loop policy evaluation. To understand these gains, we trace performance across training stages and isolate component contributions through matched ablations. We then assess whether video fidelity is maintained and whether the components also improve policy evaluation on another backbone.

## 4.1 EXPERIMENTAL SETUP

**Evaluation setting.** WorldArena benchmarks world models as interactive policy simulators (Shang et al., 2026). We evaluate four newly trained policies on its 50 RoboTwin 2.0 manipulation tasks (Chen et al., 2026), and also report results for the public benchmark's five released $\pi_{0.5}$ (Physical Intelligence et al., 2025) variants. Following WorldArena, a Qwen3-VL (Bai et al., 2025) evaluator assigns binary task success to each generated episode. We compare imagined and simulator policy success rates using Pearson correlation, Spearman correlation for ordering, and MAE and RMSE for success-rate error.

**Training data and models.** To train on different action outcomes from shared initial states, we execute the five released WorldArena policies from three simulator resets per task across all 50 tasks, collecting 750 trajectories that include both successes and failures. We reserve 15 reset groups (75 trajectories) for validation and train on the remaining 675 trajectories. Reset provenance is given in Appendix B.3.

We adapt released X-WAM on RoboTwin data to head-camera prediction at every control step, obtaining X-WAM (adapted). From this common starting point, we train three variants independently on the collected training split: X-WAM (+ data), which retains the architecture and uses only video flow matching, and the two component models described in Section 3.4. Training details appear in Appendix A.3.

## 4.2 ACTION CONSEQUENCES AND POLICY EVALUATION

**Object outcomes under action interventions.** To test whether later predictions retain the effects of earlier interactions, we briefly vary actions from a shared interaction state and then continue both branches with identical commands. We construct 24 such pairs from eight resets across five tasks, covering spatially offset grasps, delayed gripper closure, and early release. Simulator states establish that branch A holds the object while branch B leaves it behind or releases it. Using two generation seeds per pair, we recursively predict videos for 128 control steps from the shared starting state and obtain human annotations of object holding after 64 and 128 steps. Paired outcome accuracy counts a pair as correct only when both branch outcomes are clearly correct; ambiguous videos remain in the denominator (Appendix B.1).

ActionTrace achieves the highest overall paired outcome accuracy at both control steps (Figure 4(a)), exceeding X-WAM (+ data) by 6.25 percentage points. The gain is concentrated in early-release cases (Appendix B.1, Table 9), illustrated in Figure 4(b). Here, both X-WAM variants depict the red block

at the gripper in branch B after the release. ActionTrace instead preserves the distinction between the held block in A and the released block in B through step 128, matching the simulator outcomes.

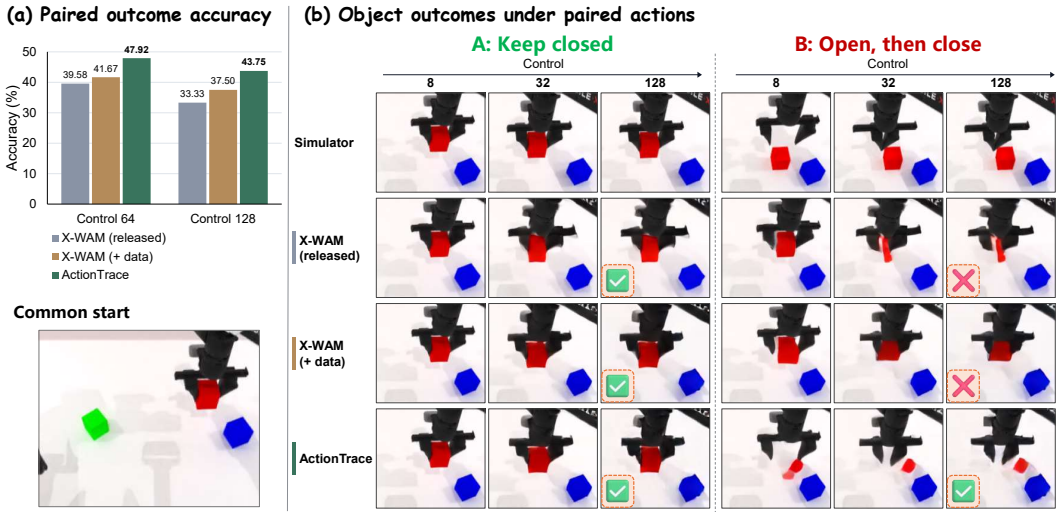


Figure 4: Object outcomes under paired actions. (a) Paired outcome accuracy across 24 action pairs and two generation seeds, requiring both branch outcomes to be correct. (b) An early-release example from a shared state with the red block held. Branch A keeps the gripper closed, while B opens and recloses it; their commands are identical after 32 control steps. The rows compare simulator frames with model predictions at control steps 8, 32, and 128, counted from the shared starting state.

**Closed-loop policy evaluation.** We next move from prescribed action pairs to closed-loop rollouts, where policies act on generated observations. Keeping the world models fixed, we evaluate four policies not used to collect world-model fine-tuning trajectories: ACT (Zhao et al., 2023) and three $\pi_{0.5}$ checkpoints, with ACT's task-specific models treated as one policy. All four share 500 new resets, with simulator executions from the same resets providing reference success rates (Appendix B.2).

ActionTrace reaches a Pearson correlation of 0.9805, compared with 0.9184 for X-WAM (+ data), and recovers the four-policy ordering (Table 1). In particular, it distinguishes the two strongest policies, which X-WAM (+ data) ranks in reverse order. The success-rate errors also decrease, with MAE falling from 0.0950 to 0.0690 and RMSE from 0.1071 to 0.0760.

Table 1: Closed-loop evaluation of four newly trained policies on 500 shared resets. Brackets give 95% paired task-bootstrap intervals for Pearson (Appendix B.2). MAE and RMSE use success rates on the $[0, 1]$ scale.

| Model | Pearson ↑ | Spearman ↑ | MAE ↓ | RMSE ↓ |
|---|---|---|---|---|
| X-WAM (released) | 0.9281 [0.8179, 0.9850] | 0.9487 | 0.1250 | 0.1417 |
| X-WAM (adapted) | 0.9143 [0.7925, 0.9769] | 0.8000 | 0.1410 | 0.1545 |
| X-WAM (+ data) | 0.9184 [0.7763, 0.9859] | 0.8000 | 0.0950 | 0.1071 |
| ActionTrace | **0.9805** [0.8917, 0.9986] | **1.0000** | **0.0690** | **0.0760** |

Table 2: WorldArena Task2 Policy Evaluator. Public baseline scores are taken from the official leaderboard.

| Model | Policy Evaluator Pearson ↑ |
|---|---|
| Cosmos-Predict 2.5 (Action) (NVIDIA et al., 2025) | 0.4830 |
| IRASim (Zhu et al., 2025b) | 0.6580 |
| RoboScape (Shang et al., 2025) | 0.8630 |
| WoW (Chi et al., 2025) | 0.9588 |
| Ctrl-World (Guo et al., 2026b) | 0.9860 |
| ActionTrace (ours) | **0.9920** |

**WorldArena benchmark comparison.** We also evaluate ActionTrace on WorldArena's public benchmark: five $\pi_{0.5}$ variants trained on 10%, 20%, 30%, 50%, and 100% of the policy data and executed from 500 shared test resets, giving 2,500 rollouts. Compared with the public WorldArena Policy Evaluator baselines (Shang et al., 2026; WorldArena Team, 2026), ActionTrace achieves the highest Pearson correlation in Table 2 (0.9920). Appendix B.3 details the rollout protocol.

### 4.3 Ablation Study

**Component analysis.** We use WorldArena's five-policy evaluation to trace model development through the training stages (Table 3). Training on the collected trajectories with video flow matching alone improves policy-level correlation modestly. Both component models achieve substantially higher correlation, with a further gain from joint training in ActionTrace.

**Component ablation.** We next isolate component contributions by matching the data split, optimization settings, and schedule at corresponding training stages (Table 4). The comparisons cover kinematic grounding and consequence conditioning, branch supervision, and cross-chunk carry. The model without branch supervision retains state supervision on recorded trajectories; Appendix A.3 gives the full configurations.

Table 3: Component analysis on WorldArena.

| Variant | $r \uparrow$ | $\rho \uparrow$ |
|---|---|---|
| X-WAM (released) | 0.7217 | 0.6000 |
| X-WAM (adapted) | 0.7172 | 0.5000 |
| + data only | 0.7616 | 0.7000 |
| + consequence conditioning | 0.9338 | 0.9000 |
| + kinematic grounding | 0.9462 | 0.9000 |
| ActionTrace | **0.9920** | **1.0000** |

Table 4: Component ablation on WorldArena.

| Variant | $r \uparrow$ | $\rho \uparrow$ |
|---|---|---|
| ActionTrace | **0.9920** | **1.0000** |
| w/o kinematic grounding | 0.9068 | 0.9000 |
| w/o consequence conditioning | 0.8936 | 0.9000 |
| w/o both components | 0.8060 | 0.8208 |
| w/o branch supervision | 0.9323 | 0.9000 |
| w/o cross-chunk carry | 0.9200 | 0.9747 |

$r$: Pearson correlation; $\rho$: Spearman correlation.

Under these matched training settings, removing either kinematic grounding or consequence conditioning lowers policy-level correlation, and removing both produces the largest drop. With both components retained, removing branch supervision also reduces correlation, supporting learning from paired action branches. Reinitializing the state between chunks, while retaining within-chunk recurrence, lowers Pearson from 0.9920 to 0.9200, showing the benefit of carrying the state across video segments.

Several intermediate and ablated models in these comparisons achieve lower MAE and RMSE than ActionTrace, but their ranking reversals or ties make them less reliable for distinguishing policy strength (Appendix B.3, Table 13). By contrast, ActionTrace recovers the complete simulator ordering with the highest Pearson correlation, despite underestimating absolute success rates on this benchmark.

**Internal ablation of consequence conditioning.** We next examine how the recurrent state, frame-aligned controls, and branch supervision contribute to consequence conditioning, ablating each during its separate training stage (Table 5; Appendix B.4). At this stage, the state recurs within each chunk and is reinitialized from the boundary image for the next; removing it also removes its latent and outcome supervision.

Table 5: Internal ablation of consequence conditioning. The final row matches + consequence conditioning in Table 3. MAE and RMSE use success rates on the $[0, 1]$ scale.

| Recurrent state | Frame-aligned controls | Branch supervision | Pearson $\uparrow$ | Spearman $\uparrow$ | MAE $\downarrow$ | RMSE $\downarrow$ |
|---|---|---|---|---|---|---|
| ✗ | ✓ | ✓ | 0.9169 | **0.9747** | 0.0834 | 0.0973 |
| ✓ | ✗ | ✓ | 0.8993 | 0.9000 | 0.0870 | 0.0990 |
| ✓ | ✓ | ✗ | 0.8558 | 0.8000 | 0.0754 | 0.0902 |
| ✓ | ✓ | ✓ | **0.9338** | 0.9000 | **0.0722** | **0.0884** |

With the recurrent state present, branch supervision raises Pearson from 0.8558 to 0.9338. Frame-aligned controls provide a complementary benefit: retaining them alongside the state and branch supervision yields the highest Pearson and lowest MAE and RMSE within this training stage.

### 4.4 Long-horizon video fidelity

To assess video fidelity alongside the gains in policy evaluation, we compare generated and simulator videos over four recursive chunks on 74 episodes from 15 held-out scenes. All chunks use recorded controls; after the first chunk, predictions are conditioned on generated boundary frames. The PSNR, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018) scores in Table 6 show that ActionTrace maintains video reconstruction quality comparable to that of X-WAM (+ data). Model-specific temporal sampling is detailed in Appendix B.5.

Table 6: Video fidelity over 64-control-step recursive rollouts.

| Model | PSNR ↑ | SSIM ↑ | LPIPS ↓ |
|---|---|---|---|
| X-WAM (released) | 19.956 | 0.8673 | 0.1362 |
| X-WAM (+ data) | 20.248 | 0.8698 | **0.1257** |
| ActionTrace | **20.329** | **0.8707** | 0.1259 |

### 4.5 Policy evaluation against real-world outcomes

We test whether ActionTrace's components improve another video backbone by transferring kinematic conditioning and the consequence state to the public Ctrl-World DROID checkpoint (Guo et al., 2026b; Khazatsky et al., 2024). Both Ctrl-World + ActionTrace components and Ctrl-World (continued), which retains the released architecture, use the same DROID data and continuation schedule. The new evaluation setups lack released camera extrinsics, so we retain a camera-independent forward-kinematics embedding and the consequence state without projected features.

We evaluate seven tasks and three policies not used to collect world-model training or continuation data. We assess 420 rollouts per model through human annotation using Ctrl-World's task-specific criteria and compare predicted and reported real-world success rates (Table 7). Relative to Ctrl-World (continued), our transferred model raises Pearson from 0.6007 to 0.7283 and reduces MAE from 0.2190 to 0.1595, also improving over the released model. Training details and the success-rate breakdown appear in Appendix B.6.

Table 7: Ctrl-World/DROID policy evaluation against real-world outcomes. Metrics compare 21 task–policy success rates. MAE and RMSE are computed from success rates on the $[0, 1]$ scale.

| Model | Pearson ↑ | Spearman ↑ | MAE ↓ | RMSE ↓ |
|---|---|---|---|---|
| Ctrl-World (released) | 0.6593 | 0.6375 | 0.1905 | 0.2309 |
| Ctrl-World (continued) | 0.6007 | 0.5828 | 0.2190 | 0.2686 |
| Ctrl-World + ActionTrace components | **0.7283** | **0.6527** | **0.1595** | **0.2124** |

## 5 Conclusion

We introduced ActionTrace, which combines spatial and temporal conditioning with same-reset branch supervision to preserve action-dependent differences in imagined robot interactions. Experiments show higher overall paired outcome accuracy and improved policy ranking with comparable video fidelity, and transferring its conditioning components to Ctrl-World also improves agreement with real-world policy outcomes. Together, these findings support explicitly learning action consequences to make imagined rollouts more informative for comparing robot policies.

**Limitations.** While ActionTrace improves policy ranking, it underestimates absolute success rates and compresses their range on WorldArena. The resulting estimates also depend on the reliability of video-based success judgments. Beyond estimation accuracy, broader validation is needed across policies and interactions: the simulation studies compare small policy sets, and the intervention study focuses on grasping and release from a limited number of initial states.

### AI USE STATEMENT

Generative AI tools were used to polish the manuscript for clarity and readability and to assist with literature search. The authors reviewed the AI-assisted revisions, verified the references against their original sources, and take responsibility for the final manuscript.

### ETHICS STATEMENT

This work uses public robot datasets and simulator-collected trajectories to study robot policy evaluation with learned world models. Before deployment, policies evaluated through imagined rollouts should undergo hardware-specific safety checks and controlled real-world validation.

### REPRODUCIBILITY STATEMENT

To support reproducibility, Section 3 describes our model and training objectives, with Algorithm 1 specifying the branch-ranking step. Appendix A provides architecture details and the full training procedure, while Section 4 and Appendix B describe the experimental protocols, data splits, and evaluation procedures. We will publicly release the code for ActionTrace.

## REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

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. In *International Conference on Machine Learning*, 2026.

Xiaowei Chi, Peidong Jia, Chun-Kai Fan, Xiaozhu Ju, Weishi Mi, Zhiyuan Qin, Kevin Zhang, Wanxin Tian, Kuangzhi Ge, Hao Li, et al. WoW: Towards a world omniscient world model through embodied interaction. *arXiv preprint arXiv:2509.22642*, 2025.

Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2786–2793, 2017.

Jun Guo, Qiwei Li, Peiyan Li, Zilong Chen, Nan Sun, Yifei Su, Heyun Wang, Yuan Zhang, Xinghang Li, and Huaping Liu. Unified 4D world action modeling from video priors with asynchronous denoising. *arXiv preprint arXiv:2604.26694*, 2026a.

Yanjiang Guo, Yucheng Hu, Jianke Zhang, Yen-Jen Wang, Xiaoyu Chen, Chaochao Lu, and Jianyu Chen. Prediction with action: Visual policy learning via joint denoising process. In *Advances in Neural Information Processing Systems*, volume 37, pp. 112386–112410, 2024.

Yanjiang Guo, Lucy Xiaoyang Shi, Jianyu Chen, and Chelsea Finn. Ctrl-World: A controllable generative world model for robot manipulation. In *International Conference on Learning Representations*, 2026b.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self Forcing: Bridging the train-test gap in autoregressive video diffusion. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, 2025.

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.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. DROID: A large-scale in-the-wild robot manipulation dataset. In *Proceedings of Robotics: Science and Systems*, 2024.

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 *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 3705–3728, 2025a.

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, Junjie Wen, Zhongyi Zhou, Yefei Chen, Chaomin Shen, Yaxin Peng, and Yichen Zhu. Scaling real-world robot policy evaluation via discrete diffusion world model. In *Forty-third International Conference on Machine Learning*, 2026.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

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

Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L. Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in Atari games. In *Advances in Neural Information Processing Systems*, 2015.

Bikang Pan, Fan Liu, Haotao Lu, Jingya Wang, and Ye Shi. SelfWAM: A self-grounded unified world action model for fast robot control. *arXiv preprint arXiv:2608.00725*, 2026.

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

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*, 2026.

Yu Shang, Xin Zhang, Yinzhou Tang, Lei Jin, Chen Gao, Wei Wu, and Yong Li. RoboScape: Physics-informed embodied world model. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, 2025.

Yu Shang, Zhuohang Li, Yiding Ma, Weikang Su, Xin Jin, Ziyou Wang, Lei Jin, Xin Zhang, Yinzhou Tang, Haisheng Su, Chen Gao, Wei Wu, Xihui Liu, Dhruv Shah, Zhaoxiang Zhang, Zhibo Chen, Jun Zhu, Yonghong Tian, Tat-Seng Chua, Wenwu Zhu, and Yong Li. WorldArena: A unified benchmark for evaluating perception and functional utility of embodied world models. *arXiv preprint arXiv:2602.08971*, 2026.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Yuang Wang, Chao Wen, Haoyu Guo, Sida Peng, Minghan Qin, Hujun Bao, Xiaowei Zhou, and Ruizhen Hu. Precise action-to-video generation through visual action prompts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12713–12724, 2025.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004.

WorldArena Team. WorldArena policy evaluator leaderboard. `https://huggingface.co/spaces/WorldArena/WorldArena`, 2026. Accessed 2026-08-31.

Zhuoyuan Wu and Jun Gao. OSCAR: Omni-embodiment action-conditioned world model for robotics. *arXiv preprint arXiv:2606.04463*, 2026.

Sherry Yang, Yilun Du, Seyed Ghasemipour, Jonathan Tompson, Leslie Kaelbling, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. In *International Conference on Learning Representations*, 2024.

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.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.016.

Zirui Zheng, Jiaqian Yu, Xiongfeng Peng, Jun Shi, Mingyi Li, Chao Zhang, Weiming Li, Dong Wang, Huchuan Lu, and Xu Jia. Mem-World: Memory-augmented action-conditioned world models for persistent robot manipulation. *arXiv preprint arXiv:2606.18960*, 2026.

Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. In *Proceedings of Robotics: Science and Systems*, 2025a.

Fangqi Zhu, Hongtao Wu, Song Guo, Yuxiao Liu, Chilam Cheang, and Tao Kong. IRASim: A fine-grained world model for robot manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9834–9844, 2025b.

# APPENDIX

## A IMPLEMENTATION DETAILS

**Video backbone.** Each 17-frame rollout chunk is encoded into one observed and four future latent positions; the observed tokens after raster addition provide the visual input $v_0$ for consequence-state initialization.

### A.1 CAMERA-ALIGNED KINEMATIC GROUNDING

Our RoboTwin implementation uses an Aloha URDF and the calibrated D435 head camera. For each 17-frame sample, the renderer receives the current 14-dimensional joint command followed by sixteen candidate future commands. It computes forward kinematics for both arms, projects link segments into the cropped head-camera image, and returns a tensor of shape $17 \times 4 \times 64 \times 80$. The first two channels contain the depth-coded arm skeletons. The other two contain circular end-effector markers with intensity encoding gripper openness. Before encoding, we sample and resize the raster to match the video's latent temporal and spatial grid. The raster encoder $\phi_r$ is a three-dimensional convolution with four input channels, 3,072 output channels, kernel size $(1, 2, 2)$, and stride $(1, 2, 2)$.

### A.2 PERSISTENT CONSEQUENCE CONDITIONING

The consequence state uses four learned queries of width 512. It aggregates each group of four future actions into one latent transition, resulting in four recurrent updates for five video latent positions. A linear projection maps the consequence state to the 3,072-dimensional video hidden width. Cross-attention and the direct state residual are inserted in the final eight of 30 video Transformer blocks. The final model uses $\alpha = 0.75$ in Eq. 8 and $\gamma = 0.125$ in Eq. 9. The outcome-prediction head is a two-layer perceptron applied to the mean terminal state and is used only during training.

**Frame-aligned control conditioning.** Encoded actions are averaged in groups of four, with a zero action feature prepended for the conditioning position. These five features are concatenated with joint-command features at RGB indices $0, 4, 8, 12, 16$. A shared, zero-initialized linear projection maps them to video width. Its outputs are broadcast spatially and added at the inputs of the final eight blocks; the original numeric tokens remain in the sequence.

### A.3 TRAINING DETAILS

**Backbone adaptation.** Starting from released X-WAM, we train on its processed RoboTwin data for 20,000 steps at learning rate $5 \times 10^{-6}$ to adapt to the head-camera view. We then train for an additional 25,000 steps at a learning rate of $3 \times 10^{-6}$ to predict at every control step, obtaining X-WAM (adapted). These stages use end-effector controls and 16-dimensional proprioceptive inputs.

**Separate component training.** From X-WAM (adapted), we train X-WAM (+ data) and the two component variants independently on the collected 750-trajectory corpus. This corpus contains 251 successes and 499 failures; 82 of its 150 reset groups contain both outcomes. We split by reset group with seed 42 into 135 training groups (675 trajectories) and 15 validation groups (75 trajectories). For all three variants, we change the numeric inputs to 14-dimensional joint-command differences and absolute joint commands, reinitializing the action and proprioceptive encoders and decoders. X-WAM (+ data) uses only the video flow-matching objective.

The consequence-conditioning variant jointly trains the recurrent state, its video cross-attention, and the frame-aligned control projection, with latent-alignment and outcome supervision. During recursive training, each new chunk initializes its state from the generated boundary image. For this variant, branch supervision pairs segments at the same time index in trajectories collected from a shared reset. Alongside ranking, it penalizes insufficient separation between the two flow predictions:

$$d = \mathrm{Mean}\left|\hat{v}^{+} - \hat{v}^{-}\right|, \qquad \mathcal{L}_{\mathrm{effect}} = [d_{\min} - d]_{+}. \tag{14}$$

The mean covers all output channels and spatiotemporal positions, including the observed position. We average this penalty over the same eligible pairs as the ranking loss, with $d_{\min} = 0.001$ and weight 0.05.

The kinematic-grounding variant uses the camera-aligned raster and branch ranking on clips beginning at the shared reset. Ranking selects pairs whose recorded futures differ by at least 0.015 in mean absolute RGB value, measured over the 32 future frames of the training window with pixels normalized to $[-1, 1]$.

The two component variants use 33-frame windows spanning two chunks with a shared boundary frame. Their future-position flow errors are weighted by motion information from the recorded video latents, and input perturbations add noise and local cutouts to the first conditioning image before video encoding. The consequence-conditioning variant uses paired-branch passes from the start; generated-context training and input perturbations ramp up over the first 2,000 steps. For the kinematic-grounding variant, branch passes ramp up from steps 2,000 to 4,000, followed by generated-context training and input perturbations from steps 6,000 to 8,000. Table 8 summarizes the training settings across stages.

Table 8: Training settings for X-WAM (+ data), the component models, and ActionTrace. Weights and probabilities are target values after their ramps. All runs use effective batch size 32 and seed 42.

| Setting | X-WAM (+ data) | Consequence conditioning | Kinematic grounding | ActionTrace (joint stage) |
|---|---|---|---|---|
| Training steps | 10,000 | 10,000 | 10,000 | 8,000 |
| Backbone learning rate | $10^{-6}$ | $10^{-6}$ | $10^{-6}$ | $2 \times 10^{-7}$ |
| Control/component LR multiplier | 10 | 10 | 10 | 10 |
| Control/component warmup (steps) | 500 | 500 | 200 | 200 |
| Backbone warmup (steps) | 500 | 500 | 200 | 250 |
| Training window length (frames) | 17 | 33 | 33 | 65 |
| Cross-chunk state carry | No | No | No | Yes |
| Motion-weighting coefficient | 0 | 1.0 | 0.5 | 0.5 |
| Latent-alignment weight | 0 | 0.10 | 0 | 0.05 |
| Outcome-prediction weight | 0 | 0.05 | 0 | 0.01 |
| Branch-ranking weight | 0 | 0.20 | 0.20 | 0.20 |
| Branch-effect penalty weight | 0 | 0.05 | 0 | 0 |
| Paired-branch pass probability | 0 | 0.50 | 0.50 | 0.75 |
| Generated-context probability | 0 | 0.75 | 0.50 | 0.50 |
| Input-perturbation probability | 0 | 0.10 | 0.10 | 0.15 |
| Initial-window oversampling factor | 1 | 1 | 32 | 8 |
| Terminal-window oversampling factor | 1 | 8 | 1 | 4 |

**Joint training of ActionTrace.** ActionTrace inherits all consequence-conditioning model parameters and copies the raster encoder's weight and bias from the kinematic-grounding variant. The new direct state residual is zero-initialized; cross-chunk carry follows Eq. 8. During the first 1,000 steps, the backbone is frozen and the raster weight $\lambda_r$ increases from zero to one. Thereafter, the final eight video blocks and the RGB output head are unfrozen with a 250-step learning-rate warmup.

Joint training uses the four-term objective in Eq. 13, with weights given in Table 8, and retains motion weighting for the video loss. Paired-branch passes ramp up from steps 1,000 to 2,000. Branch ranking uses margin $m = 0.02$ and is restricted to reset-start clips with distinct control conditions, preferentially pairing trajectories with opposite terminal outcomes. Its visual-difference threshold of 0.015 is measured over all 64 future frames of the training window.

Training windows contain 65 frames, spanning four chunks with shared boundary frames. We oversample initial and terminal windows and weight each window inversely to its repetition count. Eighteen successful trajectories shorter than 65 frames are retained by terminal padding. Within these windows, generated-context training and input perturbations ramp up from steps 4,000 to 5,000, using two or four rollout chunks. To construct the recursive context, we obtain boundary images by one-step denoising of clips from $A$'s prefix and carry their updated states forward, without gradient propagation through the prefix.

The paired pass shares this conditioning image and carried state, along with the factual noisy video input and flow time, while substituting $B$'s actions, joint commands, and raster. We compute $\mathcal{L}_{\text{flow}}$ and $\mathcal{L}_{\text{lat}}$ on the factual pass. For terminal supervision, we use the recorded success labels $y_A, y_B$ and select the two passes independently of ranking eligibility: the factual state receives $y_A$ when $A$'s

recorded clip reaches its endpoint, and the paired state receives $y_B$ when $B$'s recorded clip reaches its endpoint. We average Eq. 11 over the selected terminal samples, setting the loss to zero if none are selected. In a recursive paired pass, $y_B$ describes $B$'s recorded outcome; it need not match the outcome of executing $B$'s controls after $A$'s prefix.

**Component ablation.** The ablations of the two conditioning components in Table 4 follow the consequence-conditioning 10,000-step recipe and the subsequent 8,000-step joint recipe, retaining each stage's branch-supervision objectives. Removing consequence conditioning disables the recurrent state, its cross-attention and direct residual, cross-chunk carry, frame-aligned controls, and state-specific losses. The two variants without this component share a 10,000-step initializer trained with these paths disabled and branch supervision enabled. Variants with kinematic grounding receive the same trained raster encoder as ActionTrace. All use the corresponding stages' data split, learning rates, training windows, sampling, backbone updates, and recursive-training schedules.

The branch-supervision ablation trains the consequence-conditioning and kinematic-grounding variants without paired-branch passes, then follows the joint stage with these passes disabled. It retains the full architecture and supervises state alignment and terminal outcomes on factual samples.

The cross-chunk carry ablation starts from the same consequence-conditioning checkpoint and raster encoder as ActionTrace and retrains the 8,000-step joint stage with $\alpha = 0$. During training and evaluation, each chunk initializes its state from its boundary image. Within-chunk recurrence, state cross-attention and direct residuals, frame-aligned controls, kinematic grounding, and branch and state supervision are retained.

# B Evaluation details

## B.1 Paired object-outcome evaluation

**Action pairs.** We construct three interventions at each reset. A spatially offset grasp shifts one arm's target by 10 cm while keeping gripper timing fixed; delayed closure preserves arm commands but closes the gripper after the grasp window. Both start from the same pre-grasp state and return to identical commands after 56 control steps. Early release starts with the object already held: one branch maintains its grasp, while the other briefly opens and recloses the gripper. Their commands are identical after 32 control steps. Each pair is produced by resetting the simulator and replaying a common prefix, with agreement checked in RGB, robot and object states, and camera configuration.

Candidate scenes come from the fixed evaluation resets in Appendix B.2. We take the first eight resets in a fixed scene and task order for which all three interventions satisfy the physical and visibility criteria, with at most two resets per task. The resulting 24 pairs cover five tasks. At control steps 64 and 128, simulator object poses and contacts establish that branch A holds the target, while branch B has left it behind or released it. These outcomes must persist over the common-action continuation, and the target must remain visible within the evaluation crop.

Physical checks use windows of eight control steps. Held objects must be lifted by at least 3.5 cm, drift at most 2.5 cm relative to the gripper, and remain in contact for at least 0.10 s and 25% of the window. Missed objects rise at most 2 cm with contact below 5%; released objects fall at least 4 cm and separate by at least 4 cm in gripper-relative coordinates, also with contact below 5%. Visibility requires at least 64 target pixels per frame in the initial and scoring windows and in at least 90% of the future frames.

**Recursive prediction.** Each model receives the shared interaction-boundary RGB image, current joint commands, task instruction, and the prescribed future commands. ActionTrace initializes its consequence state from that image and carries it within each generated branch. Later chunks use their own generated boundary frames. We predict videos for 128 control steps from the shared starting state using seeds 42 and 43, 50 denoising steps, zero classifier-free guidance, and a 95% center crop at $256 \times 320$ resolution. The seed for each chunk is derived from the base seed, pair, and starting control step, matching A and B within each model.

X-WAM (+ data) and ActionTrace generate eight 16-step chunks. Released X-WAM generates four native 32-step chunks, with one future frame per four control steps; joint commands are converted to

end-effector controls by exact forward kinematics. Its initial head image is duplicated across three views, which are then generated recursively, with evaluation on the head view. Both scoring steps coincide with native output frames for every model.

**Annotation and scoring.** All 288 videos are manually annotated at control steps 64 and 128, yielding 576 judgments. Videos are randomly ordered with model and branch identities hidden. The interface shows the target in the initial image, the lossless scoring frame, and a slow-playback clip. Labels are *held*, *not held*, or *unclear*. Held means that the target is carried by the gripper; not held requires visible evidence that it remains behind or has detached. Disappearance, severe occlusion, duplication, or deformation that prevents identification is labeled unclear.

Let $\widehat{y}_{p,s,b,t}$ be the label for pair $p$, generation seed $s$, branch $b$, and control step $t$. The paired outcome accuracy is

$$\mathrm{Acc}_{\mathrm{pair}}(t) = \frac{1}{48}\sum_{p=1}^{24}\sum_{s\in\{42,43\}} \mathbf{1}[\widehat{y}_{p,s,A,t} = \text{held}]\mathbf{1}[\widehat{y}_{p,s,B,t} = \text{not held}]. \tag{15}$$

Unclear labels contribute zero; Table 9 reports the counts for each intervention.

Table 9: Correct pair–seed counts by intervention at 64 and 128 control steps. Each intervention has 16 cases (eight resets, two seeds); overall counts correspond to Figure 4(a).

| | X-WAM (released) | | X-WAM (+ data) | | ActionTrace | |
|---|---|---|---|---|---|---|
| Intervention | 64 | 128 | 64 | 128 | 64 | 128 |
| Spatially offset grasp | 7/16 | 6/16 | 3/16 | 3/16 | 4/16 | 3/16 |
| Delayed closure | 9/16 | 7/16 | 12/16 | 10/16 | 11/16 | 10/16 |
| Early release | 3/16 | 3/16 | 5/16 | 5/16 | 8/16 | 8/16 |
| Overall | 19/48 | 16/48 | 20/48 | 18/48 | 23/48 | 21/48 |

### B.2 Closed-loop policy evaluation

**Policies and scenes.** We train ACT on 50 demonstrations per task, obtaining 50 task-specific models evaluated together as one policy. Three multitask $\pi_{0.5}$ checkpoints complete the four-policy comparison: a 50-demonstration-per-task run at 5,000 and 20,000 steps, and a separate five-demonstration-per-task run at 20,000 steps. Demonstrations come from the clean RoboTwin data processed by Fast-WAM (Yuan et al., 2026), with the five-demonstration subset nested within the 50-demonstration set. These policies are fixed before world-model evaluation and are not used to collect the world-model fine-tuning trajectories.

All four policies share ten new resets per task, totaling 500 scenes. We search fixed seed ranges starting at $900000000 + 10000i$ for task index $i$, retaining scenes with successful expert demonstrations and excluding known training resets and near-duplicate initial images. The expert videos provide successful references for the video judge. Each candidate policy is separately executed from the same reset, and the simulator's task-success predicate supplies its binary reference outcome.

**Policy training.** The $\pi_{0.5}$ runs start from the public base model, using head-camera RGB and joint commands, batch size 64, seed 3407, and bfloat16. The learning rate warms up for 1,000 steps to $2.5 \times 10^{-5}$ and follows a 30,000-step cosine schedule with floor $2.5 \times 10^{-6}$. Both demonstration subsets use this schedule. Deployment uses ten denoising steps and quantile normalization.

ACT uses an ImageNet-pretrained ResNet18 encoder, hidden width 512, four encoder and seven decoder layers, and eight attention heads. It trains with batch size eight, learning rate $10^{-5}$, KL weight ten, and seed 3407; the evaluated checkpoints are from epoch 6,000. It predicts 50-action chunks without temporal aggregation.

**Closed-loop rollouts.** World models remain frozen. At each policy query, we consume 48 of the 50 predicted controls and then query again from the latest generated head image and last consumed joint command. The simulator and world-model evaluations use each task's native budget of 400–1,700 controls. Simulator execution can stop on success; generated rollouts run for the full budget. Each model produces 2,000 videos using 50 denoising steps, zero classifier-free guidance, and seed 42.

X-WAM (adapted), X-WAM (+ data), and ActionTrace consume three 16-control chunks per query. Released X-WAM consumes 32 controls followed by 16; the latter are padded to a native 32-control input, with only the first 16 controls' outputs used. Released and adapted X-WAM convert joint commands to end-effector controls through the same fixed nearest-neighbor mapping built from the original WorldArena reference arrays. X-WAM (+ data) and ActionTrace use joint commands directly.

Released X-WAM uses a 95% center crop and a duplicated initial head image for its three views; the other models use the full head image. All recursively generated views are retained for released X-WAM, and only its head view is returned to the policy and judge.

**Success-rate evaluation.** The local Qwen3-VL-32B-Instruct judge assesses each generated video using the task instruction and a successful expert demonstration from the corresponding reset. Five frames, including the first and last, are sampled uniformly from each complete reference and generated video. The judge runs in bfloat16 with tensor parallelism of one, temperature zero, and a maximum of 1,024 output tokens. Request construction, parsing, and aggregation follow the released WorldArena local evaluator.

For each policy and model, the success rate is the mean of 500 binary episode outcomes (Table 10). We compute all metrics across the resulting four policy rates.

Table 10: Success rates (%) for the four newly trained policies. Demonstration counts are per task; the two 50-demo $\pi_{0.5}$ checkpoints come from the same training run. Each entry uses 500 episodes.

| | | X-WAM | | | |
|---|---|---|---|---|---|
| Policy | Simulator | Released | Adapted | + data | ActionTrace |
| ACT (50 demos) | 7.0 | 24.8 | 23.6 | 22.4 | 18.4 |
| $\pi_{0.5}$ (50 demos, 5k steps) | 26.2 | 45.2 | 48.0 | 39.6 | 32.0 |
| $\pi_{0.5}$ (5 demos, 20k steps) | 37.8 | 48.8 | 51.4 | 42.4 | 35.2 |
| $\pi_{0.5}$ (50 demos, 20k steps) | 46.6 | 48.8 | 51.0 | 42.0 | 38.8 |

**Uncertainty across tasks.** With policies and generated rollouts fixed, we jointly resample 50 tasks with replacement for the simulator and all four world models, retaining ten episodes per policy within each sampled task. We recompute policy rates and metrics over 10,000 draws (seed 20260924), using average ranks for Spearman ties. Tables 11 and 12 give percentile 95% intervals for the metrics and paired differences from X-WAM (+ data), respectively. Across paired draws, each metric favors ActionTrace more often than X-WAM (+ data), although all four difference intervals include zero.

Table 11: 95% paired task-bootstrap intervals for the four-policy evaluation (point estimates in Table 1). MAE and RMSE use success rates on the $[0, 1]$ scale.

| Model | Pearson | Spearman | MAE | RMSE |
|---|---|---|---|---|
| X-WAM (released) | [0.8179, 0.9850] | [0.4000, 1.0000] | [0.0780, 0.2085] | [0.0912, 0.2169] |
| X-WAM (adapted) | [0.7925, 0.9769] | [0.3162, 1.0000] | [0.0775, 0.2240] | [0.0912, 0.2342] |
| X-WAM (+ data) | [0.7763, 0.9859] | [0.2000, 1.0000] | [0.0675, 0.1545] | [0.0775, 0.1703] |
| ActionTrace | [0.8917, 0.9986] | [0.8000, 1.0000] | [0.0480, 0.1165] | [0.0560, 0.1307] |

Table 12: Paired differences (ActionTrace minus X-WAM (+ data)). Better and Tied give the percentages of draws with improved and unchanged scores; higher correlations and lower errors are better. MAE and RMSE differences use the $[0, 1]$ success-rate scale.

| Metric | Observed difference | 95% interval | Better (%) | Tied (%) |
|---|---|---|---|---|
| Pearson | $+0.0621$ | $[-0.0495, +0.1929]$ | 86.47 | 0.00 |
| Spearman | $+0.2000$ | $[-0.2000, +0.6838]$ | 61.56 | 32.97 |
| MAE | $-0.0260$ | $[-0.0645, +0.0195]$ | 82.78 | 0.66 |
| RMSE | $-0.0311$ | $[-0.0671, +0.0226]$ | 85.80 | 0.01 |

### B.3 WorldArena evaluation protocol and policy-level rates

The WorldArena Clean-50 split uses 40 episodes per task for training and ten for testing. Its policy-evaluation track (Task2) maps the test examples to `episode40` through `episode49`, giving 500

distinct evaluation resets. Replaying all five policies from each reset produces 2,500 policy–reset rollouts.

Our fine-tuning corpus is collected independently in the simulator through a seed search beginning at 100000 and contains three accepted reset groups per task. Task2 provides initial observations but not the underlying simulator seeds, so we compare every evaluation initial head-camera image with the three collected resets from the same task. On the 0–255 RGB scale, known same-reset rollouts recorded under different policies have a mean absolute pixel error of 0.253–0.876, while the median distance to the nearest collected reset across all 500 evaluation observations is 8.533.

For the adapted and subsequent models, each policy query supplies 50 controls. We consume three consecutive 16-control chunks, discard the remaining two controls, and query the policy again from the latest generated frame. X-WAM (adapted) converts policy joint commands to its end-effector representation with a nearest-neighbor mapping. X-WAM (+ data), the component variants, and ActionTrace use joint commands directly. Released X-WAM retains its original temporal sampling, predicting one future frame per four control steps. Rollout length follows the WorldArena target-length rule: $\lceil 3.6L \rceil$ output frames including the initial observation, where $L$ is the length of the corresponding reference action sequence.

For the component ablations, generation uses 50 denoising steps, zero classifier-free guidance, and seed 42. These evaluations use the full input image at $256 \times 320$ resolution, whereas training uses a 95% center crop.

Success judging follows the Qwen protocol in Appendix B.2. We compare the resulting rates, each based on 500 binary judgments, with the official simulator references (Table 13).

Table 13: WorldArena success rates (%) and errors for the variants in Tables 3–5. Policy columns indicate training-data fractions; MAE and RMSE use success rates on the $[0, 1]$ scale.

| Model | 10% | 20% | 30% | 50% | 100% | MAE ↓ | RMSE ↓ |
|---|---|---|---|---|---|---|---|
| Simulator | 28.6 | 34.58 | 37.78 | 43.52 | 46.80 | — | — |
| *Component analysis* | | | | | | | |
| X-WAM (released) | 68.6 | 74.0 | 77.6 | 75.6 | 75.4 | 0.3598 | 0.3630 |
| X-WAM (adapted) | 41.0 | 44.8 | 46.4 | 44.6 | 45.8 | 0.0666 | 0.0818 |
| + data only | 31.4 | 33.2 | 35.6 | 40.0 | 35.4 | 0.0426 | 0.0560 |
| + consequence conditioning | 28.4 | 31.2 | 31.0 | 31.8 | 32.8 | 0.0722 | 0.0884 |
| + kinematic grounding | 27.8 | 30.8 | 29.8 | 32.6 | 33.2 | 0.0742 | 0.0875 |
| ActionTrace | 24.6 | 27.2 | 29.0 | 30.4 | 32.6 | 0.0950 | 0.1021 |
| *Component ablation* | | | | | | | |
| ActionTrace | 24.6 | 27.2 | 29.0 | 30.4 | 32.6 | 0.0950 | 0.1021 |
| w/o kinematic grounding | 28.2 | 30.0 | 29.4 | 30.2 | 31.0 | 0.0850 | 0.1018 |
| w/o consequence conditioning | 27.4 | 31.0 | 29.6 | 31.2 | 33.8 | 0.0766 | 0.0897 |
| w/o both components | 26.6 | 28.8 | 27.2 | 28.8 | 32.8 | 0.0942 | 0.1060 |
| w/o branch supervision | 23.2 | 29.2 | 28.8 | 30.8 | 32.2 | 0.0942 | 0.1014 |
| w/o cross-chunk carry | 25.4 | 28.6 | 30.4 | 30.4 | 31.2 | 0.0906 | 0.1016 |
| *Internal ablation of consequence conditioning* | | | | | | | |
| Consequence conditioning | 28.4 | 31.2 | 31.0 | 31.8 | 32.8 | 0.0722 | 0.0884 |
| w/o recurrent state | 27.0 | 29.4 | 30.8 | 31.2 | 31.2 | 0.0834 | 0.0973 |
| w/o frame-aligned controls | 26.0 | 28.8 | 30.6 | 31.6 | 30.8 | 0.0870 | 0.0990 |
| w/o branch supervision | 27.4 | 31.2 | 30.4 | 32.8 | 31.8 | 0.0754 | 0.0902 |

### B.4 Internal ablation of consequence conditioning

The four variants in Table 5 start from X-WAM (adapted) and follow the consequence-conditioning recipe in Appendix A.3 for 10,000 steps. At this stage, the state recurs within each chunk and is initialized from the boundary image for the next; direct state residuals and cross-chunk carry are introduced during joint training.

All variants retain the original numeric action and proprioceptive tokens. Removing the recurrent state disables its cross-attention, latent-alignment loss, and outcome-prediction loss while retaining frame-aligned controls and branch supervision. Removing frame-aligned control conditioning retains the recurrent state and its joint-command and action inputs. Removing branch supervision disables

paired passes, the branch-ranking loss, and the branch-effect penalty while retaining factual state supervision.

### B.5 Long-horizon video fidelity protocol

The fidelity evaluation uses a scene-disjoint validation split of 74 episodes from 15 held-out scenes, each with enough frames for a 64-control rollout. We divide this horizon into four consecutive 16-control chunks. The first chunk is conditioned on the recorded reset image and the next three chunks are conditioned on the preceding generated boundary frame. All chunks receive the recorded control sequence from the same simulator trajectory, and generated frames are compared only with the corresponding simulator frames from that trajectory. Conditioning frames are excluded from all metrics.

Released X-WAM is evaluated with its original temporal sampling and contributes four evaluated future frames per chunk. X-WAM (+ data) and ActionTrace predict and evaluate all sixteen future frames per chunk. For a given episode and rollout chunk, the models use the same prompt, recorded controls, target trajectory, denoising setting, and diffusion seed. PSNR, RGB SSIM, and LPIPS are computed frame-wise and then averaged over the evaluation set.

### B.6 Ctrl-World/DROID evaluation details

Both continuation models start from the same public Ctrl-World checkpoint and train for 10,000 steps on 95,273 three-view DROID episodes. They use an effective batch size of 64, backbone learning rate $2 \times 10^{-6}$, and the same optimizer, sampling schedule, seed, and checkpoint schedule. For Ctrl-World + ActionTrace components, adapter parameters use learning rate $10^{-5}$.

The transferred model uses projected Panda-arm kinematic features computed from DROID's recorded camera extrinsics and approximate ZED intrinsics. Alongside these projected features, an MLP encodes Panda link positions in the robot base frame and gripper openness into a camera-independent forward-kinematics embedding, which is added to Ctrl-World's native per-frame conditioning. A GRU summarizes the observed visual history and updates a consequence state with the control sequence; its projected states augment the same conditioning. Training combines Ctrl-World's native video denoising objective with two state-supervision losses: smooth-$L_1$ alignment of state predictions to spatially pooled future video latents (weight 0.05), and binary cross-entropy for DROID episode-success labels on terminal windows (weight 0.1).

Evaluation uses Ctrl-World's new setups outside the public DROID dataset. Their released initial conditions omit camera extrinsics, so we disable projected kinematic features while retaining the camera-independent embedding and consequence state. The three evaluation policies are not used

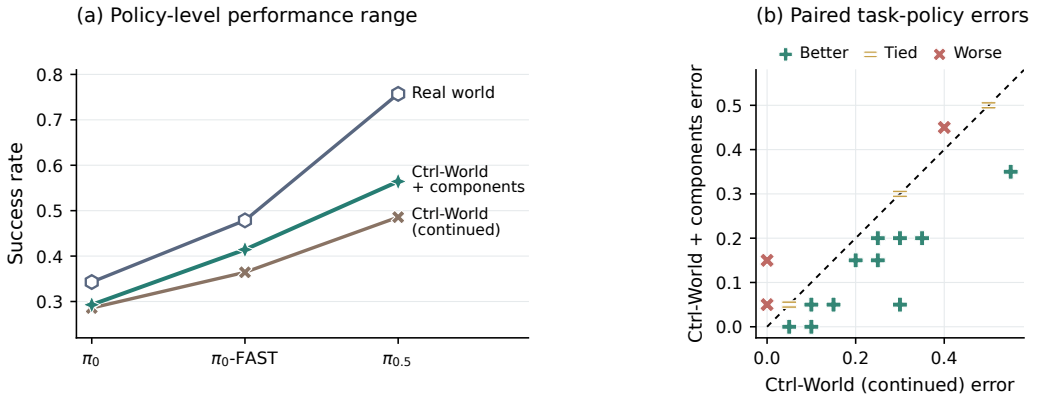


Figure 5: Ctrl-World/DROID policy-evaluation breakdown. The left panel compares policy success rates averaged across seven tasks. The right panel compares absolute errors for the 21 task–policy pairs; points below the diagonal indicate lower error for Ctrl-World + ActionTrace components than for Ctrl-World (continued).

to generate training or continuation data. Each of the 21 task–policy pairs contains 20 imagined rollouts. Task success is assessed through human annotation using Ctrl-World's task-specific criteria; real-world reference rates are also from Ctrl-World.

**Policy-level and task–policy comparison.** Figure 5 compares the two continued models. After averaging across tasks, the real-world gap between the strongest and weakest policies is 0.4143. Ctrl-World (continued) predicts a gap of 0.2000, while Ctrl-World + ActionTrace components increases it to 0.2714. At the task–policy level, Ctrl-World + ActionTrace components lowers absolute error on 14 of the 21 pairs, ties on 4, and increases it on 3. Table 14 reports the complete task–policy success rates.

Table 14: Ctrl-World/DROID success-rate breakdown. Real-world rates are reported by Ctrl-World; imagined rates are based on human annotations of 20 rollouts per task–policy pair. The final column reports results for Ctrl-World + ActionTrace components.

| | | | Ctrl-World | | |
|---|---|---|---|---|---|
| Task | Policy | Real world | Released | Continued | + components |
| Pick-Place | $\pi_0$ | 0.70 | 0.70 | 0.70 | 0.55 |
| | $\pi_0$-FAST | 0.85 | 0.60 | 0.55 | 0.65 |
| | $\pi_{0.5}$ | 0.85 | 0.55 | 0.50 | 0.65 |
| Fold-Towel | $\pi_0$ | 0.40 | 0.20 | 0.15 | 0.20 |
| | $\pi_0$-FAST | 0.55 | 0.45 | 0.45 | 0.55 |
| | $\pi_{0.5}$ | 0.75 | 0.55 | 0.45 | 0.70 |
| Drawer | $\pi_0$ | 0.05 | 0.00 | 0.00 | 0.00 |
| | $\pi_0$-FAST | 0.15 | 0.15 | 0.05 | 0.10 |
| | $\pi_{0.5}$ | 0.80 | 0.30 | 0.25 | 0.45 |
| Wipe-Table | $\pi_0$ | 0.20 | 0.65 | 0.70 | 0.70 |
| | $\pi_0$-FAST | 0.50 | 0.55 | 0.45 | 0.55 |
| | $\pi_{0.5}$ | 0.80 | 0.95 | 0.95 | 0.85 |
| Close-Laptop | $\pi_0$ | 0.15 | 0.30 | 0.10 | 0.15 |
| | $\pi_0$-FAST | 0.50 | 0.35 | 0.50 | 0.45 |
| | $\pi_{0.5}$ | 0.70 | 0.50 | 0.45 | 0.50 |
| Pull-Tissue | $\pi_0$ | 0.30 | 0.05 | 0.00 | 0.00 |
| | $\pi_0$-FAST | 0.10 | 0.15 | 0.20 | 0.10 |
| | $\pi_{0.5}$ | 0.75 | 0.40 | 0.35 | 0.30 |
| Stack | $\pi_0$ | 0.60 | 0.45 | 0.35 | 0.45 |
| | $\pi_0$-FAST | 0.70 | 0.50 | 0.35 | 0.50 |
| | $\pi_{0.5}$ | 0.65 | 0.40 | 0.45 | 0.50 |