ACID: ACTION CONSISTENCY VIA INVERSE DYNAMICS FOR PLANNING WITH WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Decision-time planning with action-conditioned world models has become a popular paradigm for embodied control. However, the standard planning cost judges a candidate solely by how close its predicted terminal state lies to the goal, leaving the realizability of the intermediate transitions unchecked—a predicted trajectory can look convincing while the environment rollout drifts away from it. In this paper, we propose ACID, a decision-time planning framework that introduces *cycle action consistency*: the action inferred backward from a predicted transition by an inverse dynamics model should recover the one that was conditioned on. We fold this per-step residual into the planning cost via a scale-invariant adaptive weight. Across four action-conditioned world models and eight tasks encompassing object manipulation and articulated control in simulation, visual navigation, alongside real-robot manipulation, ACID consistently improves planning and matches the baseline's accuracy with substantially less planning compute.

1 INTRODUCTION

World models (Ha & Schmidhuber, 2018) have become central to embodied decision making, serving as learned simulators that capture environmental dynamics within internal representations. Among the various ways of exploiting world models, decision-time planning through test-time optimization has emerged as a particularly attractive paradigm since it requires no additional policy learning and accommodates a wide variety of world model designs (Bar et al., 2025; Kim et al., 2026; Zhou et al., 2025; Maes et al., 2026; Assran et al., 2025; Sobal et al., 2025; Nam et al., 2026). A popular instantiation of this paradigm is search-based planning with an action-conditioned world model (Bar et al., 2025; Zhou et al., 2025; Hansen et al., 2024): candidate action sequences are sampled and simulated through the world model, and the trajectory that minimizes a planning cost is executed in the environment. The model predictive control (MPC) (Morari & Lee, 1999) framework with cross-entropy method (CEM) (De Boer et al., 2005) optimization follows precisely this procedure and has become a standard choice for decision-time planning with action-conditioned world models.

Despite its appeal, decision-time planning with world models relies blindly on predicted trajectories without ever checking that they are *realizable*, that is, whether each transition could actually be produced by the conditioning action in the environment. A conventional planning cost defined solely by the proximity between the goal and the terminal state cannot see this: realizability is a property of the transition, and visual plausibility at the end does not imply it. In other words, a prediction can look convincing while the actual environment rollout deviates from it. Whether the intermediate transitions are realizable, or the trajectory merely drifts toward a goal-like state, is never considered by the cost. A planner can thus commit to action sequences whose predicted goal-reaching cannot be reproduced in the environment.

Recent work has sought to close this realization gap from the world model side, *e.g.*, scaling up to large video-generative backbones (Bar et al., 2025; Kim et al., 2026; Chen et al., 2024; Du et al., 2024; Hu et al., 2023; Gao et al., 2024; Ye et al., 2026b; Bruce et al., 2024; Du et al., 2023) or enforcing action conditioning through guidance (Kim et al., 2025; Song et al., 2025). These approaches either introduce prohibitive training cost or require a specific probabilistic formulation of the world model; the latter does not apply to the recent JEPA line of work, whose predictors model latent transitions deterministically (Zhou et al., 2025; Maes et al., 2026; Assran et al., 2025; Sobal et al., 2025; Nam et al., 2026). Moreover, none addresses the planning cost: regardless of how faithful the predicted

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.

trajectories become, the objective still scores them solely by the final state proximity and cannot tell which ones genuinely realize their conditioning actions. What is needed is a complementary, decision-time mechanism that verifies action fidelity directly within the planning cost, while leaving the world model untouched and therefore composable with any of the above improvements. This raises a central question: without retraining the world model, how can the planning cost tell a trajectory that actually reaches the goal from one that only appears to?

Telling the two apart requires estimating the world model's rollout error at planning time, which cannot be done in the state space: the true outcome of a candidate action is unknown until executed. The action space, in contrast, has a known reference: the conditioning action itself. An inverse dynamics model (IDM) learns the same environment dynamics from the same transitions but reads them out in the action space, inferring the action that cause the transition between two consecutive states. Chaining the two closes a cycle, from a conditioning action through the world model and back through the IDM: if the prediction is the transition the action truly produces, the cycle closes on the conditioning action; otherwise, the rollout error surfaces as a residual between the two. This *cycle action consistency* thus turns a state-space error that cannot be measured into an action-space residual that can. Building on this principle, we introduce **A**ction **C**onsistency via **I**nverse **D**ynamics (**ACID**), a decision-time planning framework that introduces the cycle action consistency as another planning cost, preferring candidate action sequences whose predicted trajectories both reach a goal-like final state and close the cycle at every step. In doing so, we repurpose the IDM as a decision-time verifier, estimating the world model's rollout error, rather than the offline action decoder or pseudo-labeler it has conventionally been.

By augmenting decision-time planning with the cycle action consistency, ACID delivers consistent gains across four action-conditioned world models, spanning three JEPA-style latent predictors and one video generative model, and eight tasks encompassing object manipulation and articulated control in simulation, visual navigation, alongside tabletop manipulation with real robots. These gains are insensitive to hyperparameter settings and come with substantially less total planning compute than the baseline.

2 Related Work

2.1 World models

World models that learn environment dynamics and predict future states have rapidly diversified. We group prior work into three families by how each obtains executable actions. Action-conditioned world models take a current state and action as input and predict the resulting next state, and therefore rely on test-time search or optimization to find goal-reaching action sequences (Maes et al., 2026; Assran et al., 2025; Nam et al., 2026; Bar et al., 2025; Rigter et al., 2025). They span latent predictors such as JEPA-based models as well as large video backbones. State-only generative models take no action as input, but instead synthesize a goal- or language-conditioned sequence of future states and recover the action at each timestep afterward (Du et al., 2023; Xie et al., 2025). World action models instead generate future states and their corresponding actions together over a horizon, so a single forward pass emits an action sequence already aligned with the predicted visual future (Ye et al., 2026b;a; Zhu et al., 2025). This removes the need for both a separate inverse dynamics model and test-time search. Our work targets action-conditioned world models, where the action is left to test-time search and a planning cost can intervene; ACID adds its consistency cost at this decision-time stage without altering the model.

2.2 Learning inverse dynamics for robot learning

The inverse dynamics model (IDM) maps a pair of consecutive states to the action that produced the transition between them, and is widely used in robot learning. In prior work, the IDM has typically been cast in three roles, none of which influences which action sequence the planner selects: they operate after the planner has already chosen a trajectory, or only during training. As an action decoder, an IDM extracts executable actions from the output of a state-only or video-only planner. UniPi (Du et al., 2023) first generates a future video and LDP (Xie et al., 2025) first generates a latent state trajectory, after which an IDM recovers the actions; the trajectory is already chosen, and the IDM only decodes it. As an auxiliary task, inverse-dynamics prediction supplies a loss term

that shapes representations during training (Brandfonbrener et al., 2023), with no role at inference. As a pseudo-labeler, an IDM assigns action labels to otherwise action-free video, enabling policy pretraining (Baker et al., 2022; Jang et al., 2025). In contrast, we cast the IDM as a *verifier* at decision time, where its consistency signal enters the planning cost directly and shapes which action sequence the planner commits to.

3 METHOD

ACID augments decision-time planning with a per-step measure of how realizable a world model's predicted transitions are, as illustrated in Fig. 1. We first formulate world models and MPC with CEM-based planning (Sec. 3.1), introduce the cycle action consistency and the IDM that verifies it (Sec. 3.2), and fold this signal into the planning cost with a scale-invariant adaptive weight (Sec. 3.3).

3.1 PRELIMINARIES

Action-conditioned world model. An action-conditioned world model captures the environment's transition dynamics: given a current observation and an action, it predicts the resulting next observation. By chaining such one-step predictions, the model can forecast the outcome of an entire action sequence, enabling multi-step prediction. Rather than modeling these dynamics in pixel space, we primarily adopt the JEPA-style design of recent latent world models (Zhou et al., 2025; Maes et al., 2026; Sobal et al., 2025), which learn transitions in a compact latent space. Concretely, an encoder E_θ maps a raw image observation o_t to a latent state $z_t = E_\theta(o_t)$, and a world model F_θ predicts the next latent state $\hat{z}_{t+1}$ from z_t given action a_t, *i.e.*, $\hat{z}_{t+1} = F_\theta(z_t, a_t)$. The same formulation also applies to pixel-space models by taking E_θ to be the identity, which we revisit in our experiments. F_θ is trained with trajectories collected offline in advance without reward or task labels. Our work targets the goal-conditioned visual control, where the agent must reach a goal specified by an RGB image o_g. We address this task by planning action sequences through the learned world model F_θ.

Decision-time planning with action-conditioned world models. We instantiate decision-time planning as MPC over action sequences optimized with CEM. Given a current observation o_0 and a goal observation o_g, both are encoded into latent states $z_0 = E_\theta(o_0)$ and $z_g = E_\theta(o_g)$. For a candidate action sequence $a_{0:H-1}$ over a planning horizon of H steps, the world model autoregressively predicts a latent trajectory:

$$\hat{z}_{t+1} = F_\theta(\hat{z}_t, a_t), \qquad \hat{z}_0 = z_0. \tag{1}$$

Each candidate is then scored by the goal cost c_g measuring its proximity to the goal,

$$c_g(a_{0:H-1}) \;=\; \big\|\hat{z}_H - z_g\big\|_2^2. \tag{2}$$

CEM optimizes this cost by iteratively sampling action sequences from a Gaussian distribution and in turn fitting the distribution to the top-K elite candidates under c_g. Following MPC, the optimized H-step action sequence is executed in the environment, after which the next H-step action sequence is replanned using the resulting observation.

3.2 CYCLE ACTION CONSISTENCY VIA INVERSE DYNAMICS

Cycle action consistency. We introduce *cycle action consistency*, a verifier-based cost that detects when a predicted trajectory drifts from its conditioning actions: if $\hat{z}_{t+1} = F_\theta(\hat{z}_t, a_t)$ truly reflects a_t, then the action inferred from the transition $(\hat{z}_t, \hat{z}_{t+1})$ should recover a_t. As an external verifier that performs this backward inference, we introduce an IDM, denoted by G_ϕ, that infers the action responsible for a latent transition, $\hat{a}_t = G_\phi(z_t, z_{t+1})$. At each step of the predicted trajectory, we measure the *per-step consistency residual* between the conditioning action and the action recovered by the verifier: $\big\|\, a_t \;-\; G_\phi\big(\hat{z}_t,\, \hat{z}_{t+1}\big)\,\big\|_2^2$. This residual vanishes when the forward prediction $\hat{z}_{t+1} = F_\theta(\hat{z}_t, a_t)$ is the transition a_t actually produces, and grows as the predicted transition drifts from the conditioning action. The action consistency cost of a candidate sequence $a_{0:H-1}$ aggregates this residual over the planning horizon,

$$c_a(a_{0:H-1}) \;=\; \frac{1}{H}\sum_{t=0}^{H-1} \big\|\, a_t \;-\; G_\phi\big(\hat{z}_t,\, \hat{z}_{t+1}\big)\,\big\|_2^2. \tag{3}$$

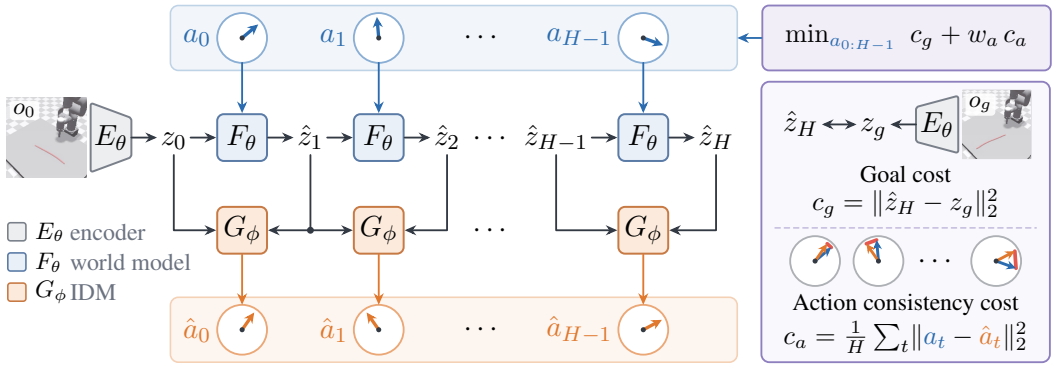

Figure 1: **Overall pipeline of ACID.** (*Left*): The encoder E_θ maps o_0 to z_0, and the world model F_θ rolls out $\hat{z}_{1:H}$ under a candidate action sequence $a_{0:H-1}$, colored in blue. The IDM G_ϕ infers from each predicted transition $(\hat{z}_t, \hat{z}_{t+1})$ the action $\hat{a}_t$, colored in orange, that explains the transition. (*Right*): CEM searches or the action sequence minimizing $c_g + w_a c_a$, where the goal cost c_g compares $\hat{z}_H$ with the encoded goal z_g, and the action consistency cost c_a averages the per-step error between a_t and $\hat{a}_t$.

This cost computation does not require additional rollout in the world model since it reuses the predicted trajectory that was used to evaluate c_g in Eq. (2).

Why the residual tracks realizability. A predicted step is *realizable* if executing a_t from $\hat{z}_t$ actually produces $\hat{z}_{t+1}$. We measure how far it is from being realizable by the *realizability error* $e_t = \hat{z}_{t+1} - f(\hat{z}_t, a_t)$, where f denotes the true latent dynamics, so that the step is realizable if and only if $e_t = 0$. To relate e_t to the per-step consistency residual, we assume locally linear latent dynamics $f(z, a) = Az + Ba$ (Wang et al., 2026) in which distinct actions produce distinct latent displacements, i.e., B has full column rank. The action that best explains a transition (z, z') is then $G^*(z, z') = B^+(z' - Az)$, where B^+ is the pseudo-inverse of B; we refer to G^* as the ideal IDM.

The realizability error splits into two parts, $e_t = e_t^{\parallel} + e_t^{\perp}$. The first part, which we call the *action-explainable error*, is an error that some action could have produced. For example, if a_t moves the agent to the right but the prediction shows it moving left, the predicted step looks exactly like the outcome of a different action. The second part is an error that no action can produce, such as an object that suddenly appears in the scene. Formally, $e_t^{\parallel}$ is the orthogonal projection of e_t onto $\mathrm{range}(B)$, the set of latent displacements that actions can cause, and $e_t^{\perp} = e_t - e_t^{\parallel}$ (Appendix A.1).

Proposition 1 (Residual as realizability error)**.** *Let $f(z, a) = Az + Ba$, where B has full column rank, and let $s_{\min}(B)$ and $s_{\max}(B)$ be the smallest and largest singular values of B. Then, for every predicted transition $\hat{z}_{t+1} = F_\theta(\hat{z}_t, a_t)$:*

(i) the residual of the ideal IDM is the realizability error mapped into the action space,

$$a_t - G^*\big(\hat{z}_t, \hat{z}_{t+1}\big) \;=\; -B^+ e_t \;=\; -B^+ e_t^{\parallel}; \tag{4}$$

(ii) the residual bounds the action-explainable error from both sides,

$$s_{\min}(B)\,\big\|B^+ e_t\big\| \;\le\; \big\|e_t^{\parallel}\big\| \;\le\; s_{\max}(B)\,\big\|B^+ e_t\big\|. \tag{5}$$

We prove Proposition 1 in Appendix A.2 and A.3. By (i), the residual is the action-explainable error translated into the action space: it is the additional action that would have been needed to produce the prediction. In the example above, the IDM recovers a leftward action, and the residual is its gap from the commanded rightward action. The action consistency cost is therefore not a heuristic that merely correlates with realizability; it measures the action-explainable part of the world model's own error. By (ii), the residual bounds the action-explainable error from both sides: a large residual implies a large error, and a small residual certifies a small one. The residual does not respond to $e_t^{\perp}$, so an object appearing from nowhere leaves it largely unchanged; for such errors, the augmented cost falls back to ranking candidates by c_g alone. Finally, the residual checks each step separately, whereas the goal cost observes only $\hat{z}_H$, into which every per-step error has already been propagated.

Algorithm 1 Decision-time Planning with Cycle Action Consistency

Require: Encoder E_θ, world model F_θ, inverse dynamics G_ϕ, current observation o_0, goal observation o_g, planning horizon H, # of samples N, # of elites K, # of iterations J, consistency weight λ

Encode states: $z_0 \leftarrow E_\theta(o_0),\ z_g \leftarrow E_\theta(o_g)$
Initialize sampling distribution $\mathcal{N}(\mu_0, \Sigma_0)$ with $\mu_0 = \mathbf{0}$, $\Sigma_0 = I$
for $j = 1$ to J **do**
  Sample N action sequences $\{a^{(n)}_{0:H-1}\}_{n=1}^{N} \sim \mathcal{N}(\mu_{j-1}, \Sigma_{j-1})$
  for $n = 1$ to N **do**
    $\hat{z}^{(n)}_0 \leftarrow z_0$
    for $t = 0$ to $H-1$ **do**
      $\hat{z}^{(n)}_{t+1} \leftarrow F_\theta(\hat{z}^{(n)}_t, a^{(n)}_t)$ ▷ World model predicted latent state
      $\hat{a}^{(n)}_t \leftarrow G_\phi(\hat{z}^{(n)}_t, \hat{z}^{(n)}_{t+1})$ ▷ IDM inferred action
    end for
    Goal cost $c^{(n)}_g \leftarrow \left\|\hat{z}^{(n)}_H - z_g\right\|_2^2$
    Action consistency cost $c^{(n)}_a \leftarrow \frac{1}{H}\sum_{t=0}^{H-1}\left\|a^{(n)}_t - \hat{a}^{(n)}_t\right\|_2^2$
  end for
  Adaptive weight $w_a \leftarrow \lambda \cdot \sigma_g/\sigma_a$, where $\sigma_g \leftarrow \mathrm{std}_n\big(c^{(n)}_g\big),\ \sigma_a \leftarrow \mathrm{std}_n\big(c^{(n)}_a\big)$
  Augmented planning cost $c^{(n)} \leftarrow c^{(n)}_g + w_a\, c^{(n)}_a$ for $n = 1, \ldots, N$
  Select the K sequences with lowest $c^{(n)}$ as elites $\mathcal{E}$: $\mu_j \leftarrow \frac{1}{K}\sum_{n\in\mathcal{E}} a^{(n)}_{0:H-1}$, $\Sigma_j \leftarrow \mathrm{Var}_{n\in\mathcal{E}}\big(a^{(n)}_{0:H-1}\big)$
end for
return optimized action sequence μ_J

Implementation of the inverse dynamics verifier. We instantiate the IDM, G_ϕ, as a flow matching (Lipman et al., 2023; Liu, 2022) action decoder built on a compact prefix-suffix transformer following π_0 (Black et al., 2024). Given two consecutive latents z_t and z_{t+1}, the model learns a velocity field $v_\phi(x_\tau, \tau, z_t, z_{t+1})$ that transports Gaussian noise $x_1 \sim \mathcal{N}(0, I)$ to the action $x_0 = a_t$ along the straight-line path $x_\tau = \tau\,\epsilon + (1-\tau)\,a_t$. Trained on real transitions, G_ϕ approximates the ideal IDM G^*; its approximation error adds a floor to the residual that, to the extent it is similar across candidates, leaves their ranking unchanged (Appendix A.5). Note that training G_ϕ requires no extra data, directly reusing training data for the world model. Also, the verifier is lightweight at inference: a single Euler step suffices in our experiments without measurable degradation, adding negligible overhead to the planning loop. We defer architectural and training details to Appendix B.

3.3 Planning with cycle action consistency

Augmented planning cost. Since the goal cost c_g alone cannot penalize a candidate action sequence whose predicted trajectory is not realizable in the environment (Sec. 3.2), we augment the planning cost with the action consistency cost c_a of Eq. (3). The augmented cost is

$$c(a_{0:H-1}) \;=\; c_g(a_{0:H-1}) \;+\; w_a \cdot c_a(a_{0:H-1}), \tag{6}$$

and we use this cost in place of c_g for CEM; the resulting procedure is summarized in Algorithm 1, with the consistency weight w_a set adaptively as will be described next. This augmented cost encourages the planner to prioritize sequences whose predicted trajectory both *reaches the goal* and *remains realizable at every step*. Under the linear model of Sec. 3.2 and with the ideal IDM, for any $w_a > 0$ the true goal distance is bounded by a constant multiple of $\sqrt{c_g + w_a c_a}$, plus a term from deviations that no action could cause (Proposition 2, Appendix A.4); minimizing the augmented cost therefore minimizes the action-explainable part of a pessimistic estimate of the true goal distance. Because the augmentation changes only the per-candidate cost, it is by construction compatible with any search-based planning optimizer (*e.g.*, gradient- or sampling-based), we instantiate it with CEM throughout though.

Scale-invariant adaptive weight. CEM ranks candidate action sequences according to their augmented costs, and selects those with the lowest costs, *i.e.*, elite candidates. Here, both the goal cost c_g and the action consistency cost c_a jointly determine the ranking of candidates, so the relative influence of the two costs decides which candidates become elites. This ranking is governed not by the absolute magnitudes of the two costs, but by how spread out their relative values across the candidates. In other words, if a cost barely vary across candidates, it will contribute little to their ranks regardless of its scale. We measured each cost's spread by its standard deviation over the

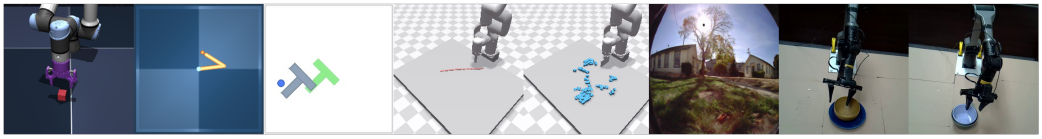

Figure 2: **Environment suites used in our experiments.** From left to right: Cube, Reacher, Push-T, Rope Manipulation, Granular Manipulation, goal-conditioned visual navigation on RECON (Shah et al., 2021), real robot tasks (PickPlace & BowlStack). More details on the environments and datasets are available in Appendix D.Results on visual navigation is presented in Appendix E.

N candidates, σ_g for c_g and σ_a for c_a, and empirically observed that it varies substantially across (1) world models, since different latent spaces produce inherently different cost distributions, (2) tasks, since the magnitude of visual change between frames differs by task and shifts the scale of the cost standard deviations, and even (3) CEM iterations, since the two spreads compress at different rates when the sampling distribution concentrates around the elites; we provide a detailed empirical analysis of these variations in Appendix H. A fixed value for w_a thus generalizes poorly as it let the impact of the action consistency cost drift in the ranking process. To ensure the two costs demonstrate comparable influence on elite selection regardless of these variations, we set the weight adaptively at every CEM iteration:

$$w_a \;=\; \lambda \cdot \frac{\sigma_g}{\sigma_a}, \tag{7}$$

which equalizes the spread of the consistency cost and the goal cost up to the factor λ, with λ controlling which term to prioritize during planning. In practice, λ is tuned once per world model and transfers across tasks without per-task retuning.

4 Experiment

Our experiments ask whether augmenting decision-time planning with the cycle action consistency improves control, how robust the gain is to the planner and to the verifier, and at what cost. We organize the study around four questions. **Q1) Generality:** Does the cycle action consistency improve planning across world models with fundamentally different latent spaces, across simulated tasks, and on a real robot (Sec. 4.2 and Sec. 4.3)? **Q2) Hyperparameter robustness:** Does the gain hold across CEM planning budgets and consistency weights λ without per-task tuning (Sec. 4.4)? **Q3) Net efficiency:** Despite the per-iteration overhead of the inverse dynamics verifier, does ACID reach a target quality with less total planning compute (Sec. 4.5)? **Q4) Robustness to verifier error:** How much error in the inverse dynamics verifier can the gain tolerate (Sec. 4.6)?

4.1 Experimental settings

Environments and tasks. We evaluate on six tasks spanning a broad range of dynamics (Fig. 2): robotic arm manipulation (Cube), articulated control (Reacher), fine-grained tabletop pushing (PushT), deformable object manipulation (Rope and Granular), and goal-conditioned visual navigation (Appendix E). For the real robot, we evaluate on two tabletop tasks (PickPlace and BowlStack).

World model baselines. We build planning on top of three JEPA-style latent predictors that differ in how their latent spaces are acquired. (1) DINO-WM (Zhou et al., 2025) learns the predictor over frozen features from a pretrained DINOv2 encoder, while (2) PLDM (Sobal et al., 2025) and (3) Le-WM (Maes et al., 2026) instead learn the encoder and predictor jointly from pixels, preventing collapse with a VICReg-family objective and a single normality regularizer, respectively. We evaluate all three world models on all five simulated tasks: Cube, Reacher, PushT, Rope, and Granular. We additionally apply ACID to a video generation-based world model, NWM trained with CompACT (Bar et al., 2025; Kim et al., 2026), on goal-conditioned visual navigation (Appendix E). For the real robot experiment, we use a separate JEPA-style latent predictor.

Evaluation. For Cube, Reacher, and PushT, we report success rate (%), the fraction of episodes whose rollout reaches the goal pose within a task-specific tolerance. For Rope and Granular, where success is not naturally binary, we instead report Chamfer distance between the achieved and target configurations. For the real robot tasks, we report the number of successful episodes out of all evaluation episodes.

Table 1: **Planning performance across world models and tasks.** The metric is success rate (%, higher is better) on Cube, Reacher, and PushT, and Chamfer distance (lower is better) on Rope and Granular. *Original* plans with the goal cost only, while *Ours* adds the action consistency cost. Values are mean ± std over three seeds.

Model	Planning cost	Cube	Reacher	PushT	Rope	Granular
		Success rate (%) ↑			Chamfer distance ↓	
Le-WM	Original	68.0 ± 5.3	75.3 ± 2.3	92.7 ± 3.1	1.79 ± 0.07	0.59 ± 0.02
	Ours	**79.3** ± 3.1	**82.0** ± 2.0	**94.0** ± 2.0	**1.26** ± 0.21	**0.43** ± 0.04
PLDM	Original	56.0 ± 2.0	75.3 ± 3.1	74.7 ± 3.1	1.95 ± 0.10	0.60 ± 0.09
	Ours	**68.0** ± 2.0	**82.7** ± 3.1	**78.7** ± 1.2	**1.38** ± 0.03	**0.49** ± 0.09
DINO-WM	Original	76.7 ± 2.3	67.3 ± 3.1	75.3 ± 6.1	1.32 ± 0.08	0.46 ± 0.02
	Ours	**81.3** ± 6.1	**68.0** ± 5.3	**76.7** ± 6.1	**0.65** ± 0.07	**0.34** ± 0.04

4.2 Planning across diverse world models and environments

Each of the three world models we consider plans with its own goal cost c_g of Eq. (2), without any modification to the planning cost. We thus adopt them as baselines and compare planning with and without the proposed action consistency cost. For every model we keep its training and the underlying CEM planner fixed, and vary only the planning cost. Throughout, *Original* plans with the goal cost only, and *Ours* adds the action consistency cost.

Planning performance. Table 1 demonstrates that incorporating the proposed action consistency cost improves planning across every world model and task. For the JEPA-style latent predictors, Le-WM and PLDM, success rate improves on all of Cube, Reacher, and PushT. On Rope and Granular manipulation tasks, Chamfer distance decreases substantially for all three world models, in particular DINO-WM on Rope. We attribute these consistent gains to a mechanism that does not depend on the task or the world model: the model can produce predicted trajectories that no action sequence could actually realize, and the action consistency cost suppresses these non-realizable candidates by checking each against the actions an inverse dynamics model infers from it. The action consistency cost modifies only the planning cost and leaves the world model untouched, so it is orthogonal to world-model improvements and can be composed with any action-conditioned world model. This extends to video generation-based world models: on goal-conditioned visual navigation with NWM trained with CompACT, the action consistency cost yields notable performance improvement (Appendix E).

Qualitative results. As shown in Fig. 3, the complex dynamics of deformable and granular media make many CEM candidate trajectories non-realizable in the actual environment, precisely the failure mode the action consistency cost is designed to cope with. For the same episode, we contrast the world model's imagined rollout with the real rollout obtained by executing the planned actions. Under the *Original* planning cost, the planned action sequence reaches the goal in the imagined rollout but drifts away from it in the real rollout, leaving the goal cost satisfied by an unreachable future. Adding the action consistency cost (*Ours*) removes this discrepancy: the real rollout closely tracks the imagined rollout and reaches the goal, confirming that the consistency term filters out the unrealizable candidates the goal cost alone cannot detect.

4.3 Real robot experiment

On a real robot, sampling joint-space action sequences from scratch is inefficient, so we adopt the policy-guided planning protocol of Chun et al. (2026), in which the world model selects among action chunks proposed by a learned policy. Since ACID changes only the per-candidate cost, it applies to this setting as is.

Table 2: **Policy guided planning performance on the real robot.** Success rate over 15 initial bowl positions per task.

Model	Planning cost	PickPlace	BowlStack
Diffusion policy	–	7/15	8/15
+World model	Original	11/15	11/15
+IDM	Ours	**14/15**	**14/15**

Setup. At each decision step, we sample $N = 20$ action chunks from the policy and predict one world-model step ahead for each; *Original* executes the candidate with the lowest c_g against a goal image, while *Ours* executes the one with the lowest augmented cost of Eq. (6) with $\lambda = 1$. The policy is a Diffusion Policy (Chi et al., 2023), the

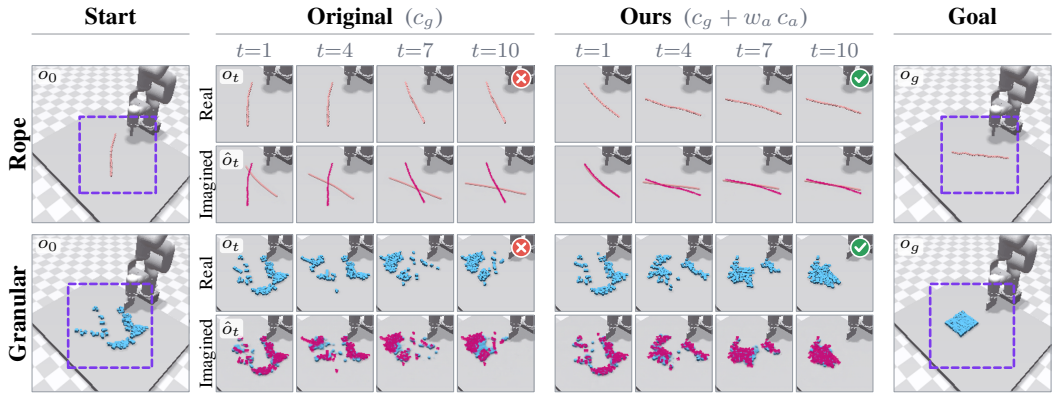

Figure 3: **Qualitative comparison on Rope and Granular with DINO-WM.** From the start observation o_0 (dashed box: zoomed region), we show the environment rollout o_t (top row) and the world model's imagined rollout $\hat{o}_t$ (bottom row) of the planned action sequence, with the real object overlaid on $\hat{o}_t$ in magenta. Under *Original*, the imagined rollout reaches the goal o_g while the real one does not; with the action consistency cost (*Ours*), the two coincide and the goal is reached.

Table 3: **CEM budget and λ sweeps on Reacher.** (*Left*): CEM budget sweep with $\lambda = 0.07$. (*Right*): λ sweep with $N = 300$.

Model	Planning cost	# CEM samples (N) 30	50	150	300	λ 0.005	0.04	0.07	0.1
Le-WM	Original	68	62	70	76		76		
	Ours	**82**	**76**	**78**	**88**	**88**	**80**	**88**	**90**
PLDM	Original	58	66	70	76		76		
	Ours	**78**	**84**	**82**	**84**	**90**	**92**	**84**	**88**

Table 4: **Constant vs. scale-invariant adaptive weight w_a on Le-WM.** Cells show the change in success rate (Δ, %) over *Original*.

Task	Original	Constant w_a 1.0	3.0	5.0	7.0	10.0	Ours
Cube	70.0	+0.0	−2.0	+0.0	+0.0	+2.0	+4.0
PushT	96.0	+0.0	+2.0	−2.0	+0.0	+2.0	+4.0
Reacher	76.0	+14.0	+6.0	−2.0	+4.0	+6.0	+12.0
Total Δ	–	+14.0	+6.0	−4.0	+4.0	+10.0	**+20.0**

world model is a JEPA-style latent predictor trained with the temporal straightening regularizer of Wang et al. (2026), and the IDM is trained on its frozen latents, all from scratch on teleoperation demonstrations. We evaluate on two tabletop tasks using a PiPER arm with a single RGB camera: PickPlace, which places a bowl on a plate, and BowlStack, which stacks one bowl on top of another. In both, the planner switches from a *pick* sub-goal to a *place* sub-goal when the gripper closes. We run one episode from each of 15 initial bowl locations unseen during training; more details are available in Appendix F.

Planning performance. As shown in Table 2, reranking the policy's samples by the goal cost alone (*Original*) raises success over a single policy sample from 7/15 to 11/15 on PickPlace and from 8/15 to 11/15 on BowlStack, and adding the action consistency cost (*Ours*) raises it further to 14/15 on both tasks. Since the policy, the world model, and the candidates are identical between the two, this gain comes from the planning cost alone, and ACID improves policy-guided planning without retraining either model.

4.4 Robustness to CEM planning budget and λ

In Table 3(left), we vary the number of CEM samples from 30 to 300 while keeping top-K at 10%. *Ours* consistently outperforms *Original* at every budget for both world models, never falling below the baseline. Table 3(right) sweeps the consistency weight λ over 4 values spanning a 20× range (0.005 to 0.1). The results indicate that a broad band of λ yields consistent gains: within each model the method does not require careful tuning of λ per task to be effective. *Ours* exhibits this level of robustness thanks to the scale-invariant adaptive weight proposed in Sec. 3.3: Table 4 isolates why the weight should be adaptive rather than constant. No single constant w_a improves all three tasks on Le-WM, and the best constant differs across tasks, which a fixed weight cannot accommodate. Our scale-invariant adaptive weight $w_a = \lambda \cdot \sigma_g / \sigma_a$, set per CEM iteration, removes this dependence and yields the largest total improvement, the robustness the sweeps above reflect.

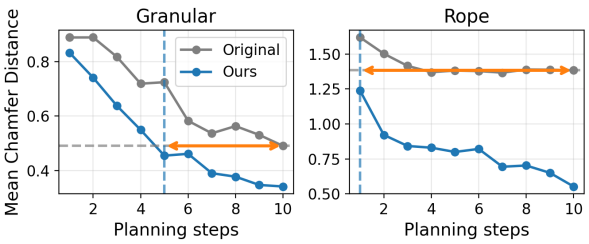

Figure 4: **Mean chamfer distance over planning steps on Granular and Rope with DINO-WM.** (lower is better). The horizontal dashed line marks the Chamfer distance that *Original* reaches at its final step, the vertical dashed line marks the step where *Ours* first reaches it.

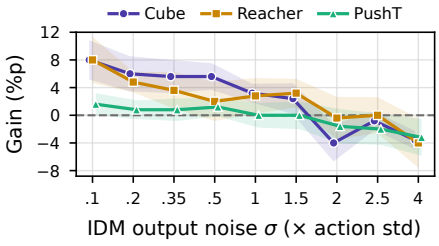

Figure 5: **Gain in success rate over *Original* under increasing IDM error, with PLDM.** σ is the injected error in units of the dataset's action standard deviation; drawn over 5 seeds per task.

4.5 Planning compute analysis

Per-iteration overhead of the inverse dynamics verifier. We measure the verifier's per-iteration overhead, defined as the ratio of the IDM's forward time to the world model's forward time within one CEM iteration. The IDM is run with a single integration step on every predicted transition of every candidate, while the world model rolls out the same candidates over the full horizon. The relative overhead is 14.5%, 14.5%, and 11.8% for Le-WM, PLDM, and DINO-WM, respectively; exact latencies are reported in Appendix I.

Lower total compute to a target quality. Since total planning compute is proportional to both the number of CEM samples and the number of planning steps, reducing either factor directly reduces the overall cost. We set the target to the quality that *Original* attains with its full budget and measure how much of that budget *Ours* requires. For Le-WM and PLDM on Reacher, the saving appears along the sample axis: in the CEM-budget sweep of Table 3, *Ours* with the smallest budget (N=30) already exceeds *Original* with the full budget (N=300), using one tenth of the samples. For DINO-WM on Granular and Rope, it appears along the planning-step axis (Fig. 4): *Ours* reaches the Chamfer distance that *Original* attains only at its final planning step in less than half the planning steps on Granular, and already after the first planning step on Rope. Thus, the samples and planning steps *Ours* saves in reaching the target more than compensate for the per-iteration overhead the verifier imposes.

4.6 Robustness to inverse dynamics verifier error

ACID uses a learned IDM rather than the oracle of our analysis, so the actions it infers carry an error; Appendix A.5 argues that this error does not systematically mask world-model errors. We test how much IDM error ACID tolerates by adding, at each replanning step, a random Gaussian offset of scale σ to the IDM's predictions, where σ is in units of the dataset's action standard deviation. We sweep σ from 0.1 to 4 with PLDM on Cube, Reacher, and PushT. As Fig. 5 shows, the gain over *Original* shrinks as σ grows, but it remains non-negative on all three tasks up to $\sigma = 1.5$, an error larger than the spread of the actions themselves, and turns negative only from $\sigma = 2$ onward.

5 Conclusion

We introduced ACID, a decision-time planning framework that augments the planning cost of action-conditioned world models with *cycle action consistency*. A separately trained inverse dynamics model maps each predicted transition back to the action space, and its residual against the conditioning action, an estimate of the world model's rollout error, is added to the standard goal cost. This consistently improves planning across diverse world models and tasks, spanning rigid and deformable object manipulation, articulated control, and visual navigation, and carries over to a real robot arm. Since the new cost changes only the per-candidate score, it is agnostic to the optimizer; while we evaluate it with sampling-based planners, extending it to gradient-based planning, and composing it with stronger world-model backbones, are natural next steps.

AI USE STATEMENT

In this work, we used generative AI tools to assist in writing proofs, to provide critical ingredients for proving mathematical claims, to propose and refine hypotheses, to provide feedback on the research methodology and experiments, to implement methods, to assist with translation, and to interpret results. We have not used generative AI tools to generate synthetic datasets, to develop theoretical models or conceptual frameworks, to formulate mathematical claims, to design the research methodology or experiments, to clean or reformat datasets, or to support qualitative and thematic data analysis. We have reviewed all AI-assisted work: all proofs, hypotheses, and interpretations of results were verified by the authors, and all AI-assisted code was checked and tested for correctness by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Mahmoud Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. In *Proc. Neural Information Processing Systems (NeurIPS)*, 2022.

Randall Balestriero and Yann LeCun. Lejepa: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*, 2025.

Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

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.

David Brandfonbrener, Ofir Nachum, and Joan Bruna. Inverse dynamics pretraining learns good representations for multitask imitation. In *Proc. Neural Information Processing Systems (NeurIPS)*, 2023.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Proc. International Conference on Machine Learning (ICML)*, 2024.

Remi Cadene, Simon Alibert, Francesco Capuano, Michel Aractingi, Adil Zouitine, Pepijn Kooijmans, Jade Choghari, Martino Russi, Caroline Pascal, Steven Palma, et al. Lerobot: An open-source library for end-to-end robot learning. In *Proc. International Conference on Learning Representations (ICLR)*, 2026.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. In *Proc. Neural Information Processing Systems (NeurIPS)*, 2024.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

Junha Chun, Youngjoon Jeong, and Taesup Kim. Sparse imagination for efficient visual world model planning. In *International Conference on Learning Representations*, volume 2026, pp. 102160–102188, 2026.

Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. In *Proc. Neural Information Processing Systems (NeurIPS)*, 2023.

Yilun Du, Mengjiao Yang, Pete Florence, Fei Xia, Ayzaan Wahid, Brian Ichter, Pierre Sermanet, Tianhe Yu, Pieter Abbeel, Joshua B Tenenbaum, et al. Video language planning. In *Proc. International Conference on Learning Representations (ICLR)*, 2024.

Shenyuan Gao, Jiazhi Yang, Li Chen, Kashyap Chitta, Yihang Qiu, Andreas Geiger, Jun Zhang, and Hongyang Li. Vista: A generalizable driving world model with high fidelity and versatile controllability. In *Proc. Neural Information Processing Systems (NeurIPS)*, 2024.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3), 2018.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *Proc. International Conference on Learning Representations (ICLR)*, 2024.

Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. Gaia-1: A generative world model for autonomous driving. *arXiv preprint arXiv:2309.17080*, 2023.

Joel Jang, Seonghyeon Ye, Zongyu Lin, Jiannan Xiang, Johan Bjorck, Yu Fang, Fengyuan Hu, Spencer Huang, Kaushil Kundalia, Yen-Chen Lin, et al. Dreamgen: Unlocking generalization in robot learning through video world models. *arXiv preprint arXiv:2505.12705*, 2025.

Dongwon Kim, Gawon Seo, Jinsung Lee, Minsu Cho, and Suha Kwak. Planning in 8 tokens: A compact discrete tokenizer for latent world model. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Seungwook Kim, Seunghyeon Lee, and Minsu Cho. Freeaction: Training-free techniques for enhanced fidelity of trajectory-to-video generation. *arXiv preprint arXiv:2509.24241*, 2025.

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

Qiang Liu. Rectified flow: A marginal preserving approach to optimal transport. *arXiv preprint arXiv:2209.14577*, 2022.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proc. International Conference on Learning Representations (ICLR)*, 2019.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworld-model: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

Manfred Morari and Jay H. Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23:667–682, 1999.

Heejeong Nam, Quentin Le Lidec, Lucas Maes, Yann LeCun, and Randall Balestriero. Causal-jepa: Learning world models through object-level latent interventions. In *Proc. International Conference on Machine Learning (ICML)*, 2026.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *Proc. International Conference on Learning Representations (ICLR)*, 2025.

Marc Rigter, Tarun Gupta, Agrin Hilmkil, and Chao Ma. AVID: Adapting video diffusion models to world models. In *Reinforcement Learning Conference*, 2025.

Dhruv Shah, Benjamin Eysenbach, Nicholas Rhinehart, and Sergey Levine. Rapid exploration for open-world navigation with latent goal models. In *Annual Conference on Robot Learning (CoRL)*, 2021.

Uladzislau Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim GJ Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. In *Proc. Neural Information Processing Systems (NeurIPS)*, 2025.

Kiwhan Song, Boyuan Chen, Max Simchowitz, Yilun Du, Russ Tedrake, and Vincent Sitzmann. History-guided video diffusion. In *Proc. International Conference on Machine Learning (ICML)*, 2025.

Jürgen Sturm, Wolfram Burgard, and Daniel Cremers. Evaluating egomotion and structure-from-motion approaches using the tum rgb-d benchmark. In *Proc. of the Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robot and Systems (IROS)*, 2012.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Ying Wang, Oumayma Bounou, Gaoyue Zhou, Randall Balestriero, Tim GJ Rudner, Yann LeCun, and Mengye Ren. Temporal straightening for latent planning. *arXiv preprint arXiv:2603.12231*, 2026.

Amber Xie, Oleh Rybkin, Dorsa Sadigh, and Chelsea Finn. Latent diffusion planning for imitation learning. In *Proc. International Conference on Machine Learning (ICML)*, 2025.

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

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026b.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. In *Proc. Neural Information Processing Systems (NeurIPS)*, 2020.

Kaifeng Zhang, Baoyu Li, Kris Hauser, and Yunzhu Li. Adaptigraph: Material-adaptive graph-based neural dynamics for robotic manipulation. *arXiv preprint arXiv:2407.07889*, 2024.

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

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. In *Proc. International Conference on Machine Learning (ICML)*, 2025.

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 *Robotics: Science and Systems (RSS)*, 2025.

APPENDIX

A THEORETICAL ANALYSIS OF CYCLE ACTION CONSISTENCY

In Sec. 3.2 we introduced the action consistency cost c_a as a signal that detects predicted transitions not realizable by their conditioning actions. Here we make this argument precise under locally linear latent dynamics. We show that (i) the residual of an ideal inverse dynamics model is exactly the world model's one-step error mapped into the action space (Sec. A.2), (ii) this residual bounds the action-explainable error from both sides (Sec. A.3), and (iii) the augmented cost of Eq. (6) upper-bounds the true distance to the goal, up to the part of the error that no action can produce (Sec. A.4). Throughout, the IDM enters only through a fixed map determined by the true dynamics and acts as an estimator of the world model's per-step rollout error.

A.1 SETTING AND NOTATION

Locally linear latent dynamics. We model the true latent dynamics f as

$$z_{t+1} \;=\; f(z_t, a_t) \;=\; Az_t + Ba_t, \tag{8}$$

where $A \in \mathbb{R}^{d\times d}$ describes the passive drift of the latent state and $B \in \mathbb{R}^{d\times d_a}$, with $d_a < d$, maps an action to the latent displacement it causes. For the nonlinear world models in our experiments, Eq. (8) should be read as the linearization of the dynamics around the predicted state, so every constant below is local. A linear latent model of this form is also adopted by Wang et al. (2026); we relax their invertibility assumption on B to the following.

Assumption 1 (Identifiability)**.** *B has full column rank.*

Assumption 1 states that distinct actions always produce distinct latent displacements, so the action between two consecutive latent states is uniquely determined. This is the property ACID exploits throughout (Sec. 1). It concerns how actions register in the latent space rather than whether the dynamics are deterministic (Sec. A.5).

Pseudo-inverse and singular values. We denote by $s_{\max}(B) \geq s_{\min}(B)$ the largest and smallest singular values of B; we write s rather than σ to avoid confusion with the cost standard deviations σ_g and σ_a of Eq. (7). Assumption 1 is equivalent to $s_{\min}(B) > 0$. Under this assumption, the Moore–Penrose pseudo-inverse of B is $B^+ = (B^\top B)^{-1}B^\top \in \mathbb{R}^{d_a\times d}$, and $P = BB^+$ is the orthogonal projection onto $\mathrm{range}(B)$, the subspace of latent displacements that some action can produce. We use the following standard facts.

Lemma 1. *Under Assumption 1, the following hold.*

(a) $B^+B = I_{d_a}$.

(b) $BB^+ = P$.

(c) $B^+ = B^+P$, *so that* $B^+v = 0$ *for every* $v \perp \mathrm{range}(B)$.

(d) $s_{\min}(B)\,\|x\| \leq \|Bx\| \leq s_{\max}(B)\,\|x\|$ *for every* $x \in \mathbb{R}^{d_a}$.

Ideal inverse dynamics model. Under Eq. (8), the action that explains a transition (z, z') in the least-squares sense is

$$G^*(z, z') \;=\; \arg\min_a \big\|z' - Az - Ba\big\|_2^2 \;=\; B^+\big(z' - Az\big). \tag{9}$$

On every real transition, $z_{t+1} - Az_t = Ba_t$, so Lemma 1(a) gives $G^*(z_t, z_{t+1}) = B^+Ba_t = a_t$: the ideal IDM recovers the executed action exactly. We refer to G^* as the oracle IDM. The learned verifier G_ϕ approximates it on the transitions of the offline dataset, and we account for its approximation error in Sec. A.5.

Realizability error. We measure how far a predicted transition is from being realizable by comparing it with the transition that its conditioning action actually produces from the same state.

Definition 1 (Realizability error)**.** *For a predicted transition $\hat{z}_{t+1} = F_\theta(\hat{z}_t, a_t)$, the realizability error is*

$$e_t \;=\; \hat{z}_{t+1} - f(\hat{z}_t, a_t) \;=\; \hat{z}_{t+1} - A\hat{z}_t - Ba_t. \tag{10}$$

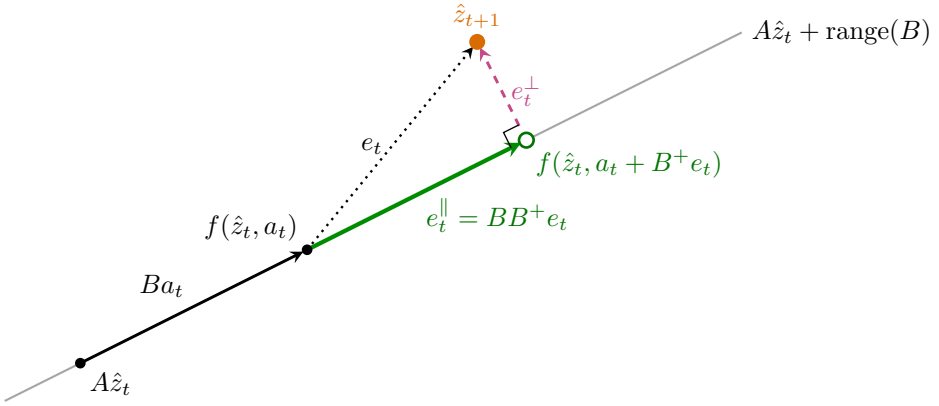

Figure 6: **Geometry of the realizability error.** Dots are latent states and arrows are displacements. From $\hat{z}_t$, the passive drift reaches $A\hat{z}_t$, and the states that some action can reach form the line $A\hat{z}_t + \mathrm{range}(B)$. The conditioning action a_t reaches $f(\hat{z}_t, a_t) = A\hat{z}_t + Ba_t$, whereas the world model predicts $\hat{z}_{t+1}$, which is off by e_t. The oracle IDM projects $\hat{z}_{t+1}$ onto the line and reads off the action $a_t + B^+e_t$, which reaches the foot of the projection (Proposition 1): the component $e_t^{\parallel}$ shifts the recovered action, whereas $e_t^{\perp}$ is discarded.

The realizability error e_t is the world model's one-step error at its own input: the predicted transition is realizable by its conditioning action if and only if $e_t = 0$. Because e_t is measured from the predicted state $\hat{z}_t$, it excludes the drift accumulated at earlier steps, which is already contained in $\hat{z}_t$. We decompose it as $e_t = e_t^{\parallel} + e_t^{\perp}$, where $e_t^{\parallel} = Pe_t \in \mathrm{range}(B)$ and $e_t^{\perp} = (I - P)\, e_t \perp \mathrm{range}(B)$. The component $e_t^{\parallel}$ is a deviation that some other action could have produced, whereas $e_t^{\perp}$ is a deviation that no action can produce. Fig. 6 illustrates this decomposition.

A.2 The residual is the world model's error in the action space

Proposition 1 in Sec. 3.2 states two properties of the residual, which we prove in this and the next subsection. Its full-column-rank condition is Assumption 1, and G^* is the ideal IDM of Eq. (9).

Proof of Proposition 1(i). By Eq. (9), Eq. (10), and Lemma 1(a), $G^*(\hat{z}_t, \hat{z}_{t+1}) = B^+(\hat{z}_{t+1} - A\hat{z}_t) = B^+(Ba_t + e_t) = a_t + B^+e_t$, which rearranges to the first equality of Eq. (4). The second follows from Lemma 1(c). □

Substituting Eq. (4) into Eq. (3) shows that, with the ideal IDM, the action consistency cost is

$$c_a(a_{0:H-1}) \;=\; \frac{1}{H}\sum_{t=0}^{H-1} \left\| B^+e_t \right\|_2^2. \tag{11}$$

Proposition 1(i) states that the IDM reports the conditioning action shifted by $\Delta a_t = B^+e_t$, the action-space image of the world model's error; Δa_t is the additional action that would explain the predicted displacement. The residual therefore measures the world model's own one-step error, and the IDM acts as an estimator of this error through the fixed map B^+, a function of the true dynamics. Moreover, B^+e_t depends only on the error made at step t and does not compound along the horizon, whereas the goal cost c_g observes only $\hat{z}_H$, into which the world model has already propagated every $e_0, \ldots, e_{H-1}$ (Sec. A.4).

By Lemma 1(c), $B^+e_t = B^+e_t^{\parallel}$. The residual thus vanishes if and only if $e_t \perp \mathrm{range}(B)$: it vanishes whenever the prediction is realizable ($e_t = 0$), but it does not detect a deviation that lies entirely outside $\mathrm{range}(B)$. We return to this blind spot in Sec. A.4.

A.3 THE RESIDUAL IS A TWO-SIDED PROXY OF THE ACTIONABLE ERROR

The residual is measured in the action space, while the error it reflects lives in the latent space. Proposition 1(ii) relates their magnitudes.

Proof of Proposition 1(ii). By Lemma 1(b), $e_t^{\parallel} = Pe_t = BB^{+}e_t = B\,\Delta a_t$. Applying Lemma 1(d) with $x = \Delta a_t$ gives $s_{\min}(B)\,\|\Delta a_t\| \le \|e_t^{\parallel}\| \le s_{\max}(B)\,\|\Delta a_t\|$, which is Eq. (5). □

The two inequalities of Eq. (5) play different roles. The right inequality, $\|e_t^{\parallel}\| \le s_{\max}(B)\,\|B^{+}e_t\|$, states that the scaled residual never underestimates the action-explainable error: a small residual certifies a small $e_t^{\parallel}$, so such an error cannot escape the cost. Pessimistic model-based reinforcement learning requires exactly such an upper bound on the model's error (Yu et al., 2020), and here it holds by construction. The left inequality, $s_{\min}(B)\,\|B^{+}e_t\| \le \|e_t^{\parallel}\|$, states that the residual cannot be large unless the error is, so an accurate prediction is never penalized heavily. It relies on Assumption 1 and weakens as $s_{\min}(B) \to 0$. Both bounds are attained, for errors aligned with the left singular vectors of B associated with $s_{\max}(B)$ and $s_{\min}(B)$, respectively.

A.4 THE AUGMENTED COST BOUNDS THE TRUE GOAL DISTANCE

The goal cost c_g of Eq. (2) scores a candidate by its predicted terminal state alone. We now relate the augmented cost of Eq. (6) to the terminal state z_H reached when the same action sequence is executed from z_0 under the true dynamics. Let $\delta_t = \hat{z}_t - z_t$ denote the drift of the predicted trajectory from the true one, with $\delta_0 = 0$. Subtracting $z_{t+1} = Az_t + Ba_t$ from $\hat{z}_{t+1} = A\hat{z}_t + Ba_t + e_t$ gives

$$\delta_{t+1} \;=\; A\,\delta_t + e_t, \qquad \text{hence} \qquad \delta_H \;=\; \sum_{t=0}^{H-1} A^{H-1-t}\,e_t. \tag{12}$$

Unlike the residual, the drift accumulates every per-step error, amplified by the passive dynamics.

Proposition 2 (Goal-distance bound)**.** *Let $\rho = \|A\|_2$. Under Assumption 1, with the oracle IDM, every candidate $a_{0:H-1}$ satisfies*

$$\|z_H - z_g\| \;\le\; \sqrt{c_g} \;+\; s_{\max}(B) \sum_{t=0}^{H-1} \rho^{\,H-1-t}\,\big\|B^{+}e_t\big\| \;+\; \sum_{t=0}^{H-1} \rho^{\,H-1-t}\,\big\|e_t^{\perp}\big\|. \tag{13}$$

Moreover, for every weight $w > 0$,

$$\|z_H - z_g\| \;\le\; \sqrt{\Big(1 + \frac{\nu}{w}\Big)\big(c_g + w\,c_a\big)} \;+\; \sum_{t=0}^{H-1} \rho^{\,H-1-t}\,\big\|e_t^{\perp}\big\|, \tag{14}$$

where $\nu = s_{\max}(B)^2 H\,\Gamma_H$ and $\Gamma_H = \sum_{k=0}^{H-1} \rho^{\,2k}$.

Proof. Eq. (13). Since $z_H = \hat{z}_H - \delta_H$,

$$\begin{aligned}
\|z_H - z_g\| \;&\le\; \|\hat{z}_H - z_g\| + \|\delta_H\| \\
&\le\; \sqrt{c_g} \;+\; \sum_{t=0}^{H-1} \rho^{\,H-1-t}\|e_t\| \\
&\le\; \sqrt{c_g} \;+\; \sum_{t=0}^{H-1} \rho^{\,H-1-t}\Big(\big\|e_t^{\parallel}\big\| + \big\|e_t^{\perp}\big\|\Big) \\
&\le\; \sqrt{c_g} \;+\; s_{\max}(B) \sum_{t=0}^{H-1} \rho^{\,H-1-t}\big\|B^{+}e_t\big\| \;+\; \sum_{t=0}^{H-1} \rho^{\,H-1-t}\big\|e_t^{\perp}\big\|.
\end{aligned}$$

The first inequality is the triangle inequality; the second follows from Eq. (12) and $\|A^{H-1-t}\| \le \rho^{\,H-1-t}$; the third splits $e_t = e_t^{\parallel} + e_t^{\perp}$; and the last applies the right inequality of Eq. (5), $\|e_t^{\parallel}\| \le s_{\max}(B)\|B^{+}e_t\|$.

Eq. (14). Starting from Eq. (13),

$$\begin{aligned}\|z_H - z_g\| \;&\le\; \sqrt{c_g} \;+\; s_{\max}(B) \sum_{t=0}^{H-1} \rho^{\,H-1-t} \big\|B^+ e_t\big\| \;+\; \sum_{t=0}^{H-1} \rho^{\,H-1-t} \big\|e_t^{\perp}\big\| \\ &\le\; \sqrt{c_g} \;+\; \sqrt{\nu\, c_a} \;+\; \sum_{t=0}^{H-1} \rho^{\,H-1-t} \big\|e_t^{\perp}\big\| \\ &\le\; \sqrt{\Big(1 + \frac{\nu}{w}\Big)\big(c_g + w\, c_a\big)} \;+\; \sum_{t=0}^{H-1} \rho^{\,H-1-t} \big\|e_t^{\perp}\big\|.\end{aligned}$$

The second inequality applies the Cauchy–Schwarz inequality with Eq. (11): $\sum_t \rho^{\,H-1-t}\|B^+ e_t\| \le \big(\sum_t \rho^{\,2(H-1-t)}\big)^{1/2}\big(\sum_t \|B^+ e_t\|^2\big)^{1/2} = \sqrt{\Gamma_H}\,\sqrt{H c_a}$, and $s_{\max}(B)\sqrt{\Gamma_H H c_a} = \sqrt{\nu\, c_a}$. The third applies Young's inequality $2xy \le \mu x^2 + y^2/\mu$, which gives $(x+y)^2 \le (1+\mu)\,x^2 + (1+1/\mu)\,y^2$ for any $\mu > 0$. With $x = \sqrt{c_g}$, $y = \sqrt{\nu\, c_a}$, and $\mu = \nu/w$, the right-hand side is $(1+\mu)\,c_g + (1 + 1/\mu)\,\nu\, c_a = (1+\mu)(c_g + w\, c_a)$, since $(1 + 1/\mu)\,\nu = \nu + w = (1+\mu)\,w$. □

Interpretation. A candidate whose predicted trajectory drifts toward the goal through world-model error can attain a small c_g while its true terminal state remains far from the goal, and the goal cost alone cannot detect this. Proposition 2 shows that the augmented cost closes this gap for the action-explainable part of the drift: for any weight $w_a > 0$, the true goal distance is bounded by the square root of the augmented cost $c_g + w_a c_a$ of Eq. (6), scaled by $\sqrt{1 + \nu/w_a}$, plus the deviations that no action can produce. Since this factor depends only on w_a, A, B, and H, it is common to all candidates of a CEM population, so ranking candidates by the augmented cost ranks them by the action-explainable part of this upper bound; the remaining term, which neither cost observes, is discussed below. Minimizing the augmented cost therefore minimizes the action-explainable part of a pessimistic estimate of the true goal distance, analogous to the uncertainty-penalized reward of MOPO (Yu et al., 2020).

Relation to the adaptive weight. Eq. (14) holds for every $w > 0$, so the adaptive weight of Eq. (7) does not affect the validity of the bound; w_a only sets how tight it is. The choice of w_a itself is governed by the relative spread of the two costs across candidates (Sec. 3.3 and Sec. H).

The component outside $\mathrm{range}(B)$**.** The last term of Eq. (13) is not controlled by c_a: by Lemma 1(c), the ideal IDM discards $e_t^{\perp}$. We keep this term rather than assuming it away. The action consistency cost thus bounds only the *action-explainable* part of the drift, that is, deviations that could have been caused by some action: two candidates whose predictions differ only in $e_t^{\perp}$ receive the same c_a, and the augmented cost orders them by c_g alone, as in the baseline.

A.5 Learned inverse dynamics model and stochastic dynamics

Learned IDM. In practice we use a learned verifier $G_\phi = G^* + \eta$, where $\eta(z, z')$ is its approximation error. Writing $\eta_t = \eta(\hat{z}_t, \hat{z}_{t+1})$, the per-step residual becomes $\|a_t - G_\phi(\hat{z}_t, \hat{z}_{t+1})\|^2 = \|B^+ e_t + \eta_t\|^2$, which expands as

$$\big\|B^+ e_t + \eta_t\big\|^2 \;=\; \underbrace{\big\|B^+ e_t\big\|^2}_{\text{world-model error}} \;+\; \underbrace{\|\eta_t\|^2}_{\text{IDM floor}} \;+\; \underbrace{2\,\big\langle B^+ e_t,\, \eta_t \big\rangle}_{\text{cross term}}. \tag{15}$$

The first term ranks candidates by the world model's error as analyzed above. The second term is an approximation floor of the IDM; to the extent that it is similar across the candidates of a population, it acts as a common offset and leaves the ranking unchanged, since only differences across candidates move the ordering (Sec. H). The cross term is the only way the IDM could mask a world-model error, and it would require η_t to be systematically anti-aligned with $B^+ e_t$ across candidates. G_ϕ is trained on real transitions, independently of F_θ, to predict actions rather than states, and no loss couples the two models; nothing in either objective induces such an alignment. We note that this is an assumption supported by independent training rather than a guarantee. Sec. 4.6 tests it empirically by adding a random offset to the predictions of a trained IDM: the gain over *Original* remains non-negative up to an offset of $1.5\times$ the spread of the dataset's actions.

Identifiability, not determinism. Assumption 1 constrains how actions register in the latent space, not whether transitions are deterministic. With additive noise, $z_{t+1} = Az_t + Ba_t + \xi_t$ with ξ_t independent of a_t, the least-squares action of a real transition is $B^+(z_{t+1} - Az_t) = a_t + B^+\xi_t$. It deviates from a_t by $B^+\xi_t$, whose distribution does not depend on the executed action, so the action remains identifiable up to a noise level set by ξ_t and $s_{\min}(B)$. Proposition 1 continues to hold with f taken as the noise-free part of the dynamics, and Proposition 2 holds with an additional term $\sum_{t=0}^{H-1} \rho^{H-1-t}\|\xi_t\|$ on its right-hand side, which does not depend on the candidate and thus leaves the ranking unchanged. Consistent with this, the action consistency cost still improves planning on goal-conditioned visual navigation with NWM (Sec. E), whose world model and IDM are trained on real-world trajectories with inherently stochastic transitions rather than on noise-free ones.

B Implementation Details of IDM

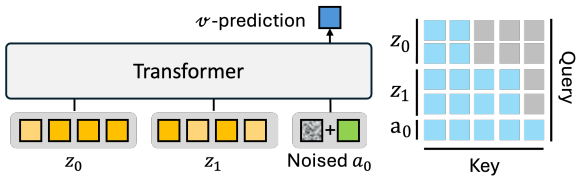

Figure 7: **Inverse Dynamics Model(IDM) architecture.** (*Left*): the prefix–suffix transformer that predicts the action between the latents of two consecutive frames z_t and z_{t+1}. (*Right*): the corresponding attention mask.

Architecture. In Fig. 7, we illustrate the IDM, including its model architecture and attention mask. The IDM is a 4-layer transformer with 3 heads and a width of 192. The latents z_t and z_{t+1} form the prefix, and a single suffix token carries the noisy action x_τ. The prefix attends only to itself, while the suffix attends to both prefix tokens; the suffix output is linearly mapped to the velocity in the action space. We use a flow matching objective (Lipman et al., 2023; Liu, 2022) with timesteps drawn from a $\mathrm{Beta}(1.5, 1.0)$ schedule.

Training data. We train a separate IDM for each task, using that task's offline dataset and the frozen encoder latents of the corresponding world model. Each example is a transition (z_t, z_{t+1}, a_t), where z_t, z_{t+1} are the encoder latents of two consecutive frames and a_t is the action between them; transitions are formed only within episode boundaries. Actions are standardized to zero mean and unit variance using per-dimension training-set statistics, and de-normalized at inference. We hold out 10% of the transitions as a validation split.

Training and inference. IDM is trained for 200K steps with batch size 256. We use AdamW (Loshchilov & Hutter, 2019) with peak learning rate 10^{-4} ($\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay 10^{-4}), with linear warmup followed by cosine decay. During inference, we integrate the learned velocity field with a single Euler step for Push-T, Reacher, OGBench-Cube, Rope manipulation, and Granular manipulation.

C World Model Baselines

C.1 Model Descriptions

We conduct experiment with three JEPA-style latent predictors that differ in how their latent spaces are acquired. They share the same structure—an encoder maps a frame observation to a latent, and a predictor predicts the next-frame latent from the current latent and action—and differ mainly in how the latent space is obtained and how representation collapse is prevented.

Le-WM (Maes et al., 2026) learns the encoder and predictor jointly from pixels, with a ViT encoder and a transformer predictor that receives the action through adaptive layer normalization. It prevents collapse with a single regularizer, SIGReg (Balestriero & LeCun, 2025), which encourages the latent embeddings to follow an isotropic Gaussian by applying a normality test along random one-dimensional projections, in place of the multiple regularizers used by prior end-to-end approaches.

PLDM (Sobal et al., 2025) also trains the encoder and predictor jointly from pixels, optionally using an ensemble of predictors. It prevents collapse with a VICReg-family objective that combines variance and covariance regularization with temporal regularization and an inverse dynamics modeling (IDM) term, so its training objective is composed of several regularizers.

DINO-WM (Zhou et al., 2025) instead learns only the predictor over frozen features from a pretrained DINOv2 encoder, sidestepping collapse through the fixed pretrained representation; we disable its non-visual (e.g., proprioceptive) inputs so that all methods observe only pixels unless otherwise noted.

C.2 Training configuration

PLDM & Le-WM. For PushT, Reacher, and OGBench-Cube, we use the official pretrained Le-WM checkpoints released by the authors (Maes et al., 2026), and train PLDM with the codebase and hyperparameters reported in the Le-WM paper. Since neither the original Le-WM nor PLDM paper evaluates on the Rope and Granular tasks, we add these tasks ourselves and train both models on the dataset released by DINO-WM (Zhou et al., 2025), following the same training configuration as DINO-WM.

DINO-WM. For OGBench-Cube, Reacher, and PushT, we train DINO-WM for as many epochs as needed to reach the performance reported in Maes et al. (2026). For the Rope and Granular tasks, we follow the training configuration and official codebase of the original DINO-WM paper (Zhou et al., 2025) without modification.

The dataset details and training configurations of all three world models are summarized in Tab. 5.

Table 5: **Dataset details and world model training configurations.** *Size* is the number of trajectories and *# Frames* the trajectory length. History length, frame skip, and train epochs are for training the world model; history length and frame skip are shared across the three world models. For Le-WM on PushT, Reacher, and OGBench-Cube, we use the official pretrained checkpoints, and the listed epochs are those used to train them.

					Train epoch		
Dataset	Size	# Frames	History len	Frame skip	Le-WM	PLDM	DINO-WM
PushT	18685	125 (avg.)	3	5	10	10	50
Reacher	10000	201	3	5	10	10	50
OGBench-Cube	10000	201	3	5	10	10	10
Rope	1000	20	1	1	100	100	100
Granular	1000	20	1	1	100	100	100

D Experimental settings

D.1 Environments and Tasks.

We evaluate our method across five continuous-control benchmarks that span a spectrum of difficulty, ranging from rigid and deformable object manipulation to articulated control; goal-conditioned visual navigation with NWM is described in Appendix E. Each task poses distinct challenges for world-model-based planning, from reasoning about contact dynamics to coordinating deformable object. The specifics of each environment and its associated dataset are described below.

Push-T is a contact-rich 2D manipulation task in which a controllable agent, depicted as a blue circle, must push a green T-shaped block until it aligns with a prescribed target configuration marked in gray. The agent interacts with the block solely through pushing, and successful completion demands an accurate model of the contact dynamics arising between the agent and the object—a property that makes this task a demanding benchmark for object manipulation. We adopt the dataset and protocol of Le-WM (Maes et al., 2026), comprising 18,685 expert episodes whose trajectories average 125 steps in length.

Reacher is a continuous-control task drawn from the DeepMind Control Suite (Tassa et al., 2018), in which a two-jointed robotic arm must be actuated so that it reaches a designated target within a 2D plane. Following DINO-WM (Zhou et al., 2025), we adopt the more stringent variant in which success requires the full arm configuration—rather than merely the end-effector—to align precisely with the goal pose. The training dataset consists of 10,000 episodes of 200 steps each, gathered using a Soft Actor-Critic policy.

OGBench-Cube is a continuous 3D robotic manipulation task originally proposed by OGBench (Park et al., 2025), in which a robotic arm equipped with an end-effector must grasp a cube and relocate it to a specified target location. Unlike the planar tasks above, this environment requires reasoning over full 3D spatial geometry as well as the coordinated sequence of grasping and placing, making it the most challenging of the three benchmarks we consider. We restrict our attention to the single-cube variant. Trajectories are produced using the data-collection heuristic supplied by the benchmark library, yielding 10,000 episodes of 200 steps each.

Rope Manipulation is a deformable-object manipulation task originally introduced by AdaptiGraph (Zhang et al., 2024), in which an XArm interacts with a soft rope placed on a tabletop, simulated in Nvidia Flex. The objective is to manipulate the rope from an arbitrary initial configuration toward a goal configuration specified at test time. Unlike rigid-object tasks, this setting probes whether the world model can faithfully capture the deformable dynamics of the rope together with the contact behavior induced by the robot's pushing actions. We use the dataset released by DINO-WM (Zhou et al., 2025), which consists of 1,000 trajectories, each spanning 20 time steps of random actions initiated from random starting positions, while testing uses goal configurations drawn from a range of initial states with random perturbations in orientation and spatial placement.

Granular Manipulation is a multi-particle manipulation task built on the same simulation backend as the rope task, in which the robot must reshape a collection of roughly one hundred granular particles into a desired target arrangement. Because the particles are disjoint and respond independently to direct contact, this environment stresses the model's capacity to predict the collective dynamics of many loosely coupled objects under physical interaction. We use the dataset released by DINO-WM (Zhou et al., 2025), which consists of 1,000 trajectories of 20 time steps of random actions, all launched from a common initial configuration, while testing is performed on specific target shapes from diverse starting states with random variation in particle distribution, spacing, and orientation.

D.2 Details of evaluation protocol

For PushT, OGBench-Cube, and Reacher, we follow the goal-conditioned control evaluation protocol of the original Le-WM paper (Maes et al., 2026) without modification for all three world models. Each evaluation episode is defined by two quantities: the evaluation budget, i.e., the maximum number of actions the agent may execute in the environment, and the goal distance, i.e., how many timesteps in the future the goal state is sampled relative to the initial state. The initial state is obtained by randomly sampling a state from a trajectory in the offline dataset, and the goal state is the state occurring a fixed number of timesteps later in the same trajectory, which ensures that the goal is reachable and consistent with the dataset dynamics. In all three tasks, the evaluation budget is 50 steps and the goal is sampled 25 timesteps in the future.

For the Rope and Granular manipulation tasks, we likewise follow the evaluation protocol of the original DINO-WM paper (Zhou et al., 2025) without modification for all three world models. In both tasks, the agent must push a deformable object into a target configuration, and performance is measured by the Chamfer Distance (CD) between the achieved and goal states rather than a binary success rate. Due to the high environment stepping cost of these two environments, each is evaluated on 10 instances. For granular manipulation, each instance samples a random initial configuration from the validation set, with the goal of pushing the material into a square shape at a randomly selected location and scale.

Algorithm 2 Decision-time Planning with CEM

Require: Encoder E_θ, world model F_θ, current observation o_0, goal observation o_g, planning horizon H, # of samples N, # of elites K, # of iterations J

Encode states: $z_0 \leftarrow E_\theta(o_0),\ z_g \leftarrow E_\theta(o_g)$
Initialize sampling distribution $\mathcal{N}(\mu_0, \Sigma_0)$ with $\mu_0 = \mathbf{0}$, $\Sigma_0 = I$
for $j = 1$ to J **do**
  Sample N action sequences $\{a^{(n)}_{0:H-1}\}_{n=1}^{N} \sim \mathcal{N}(\mu_{j-1}, \Sigma_{j-1})$
  for $n = 1$ to N **do**
    $\hat{z}^{(n)}_0 \leftarrow z_0$
    for $t = 0$ to $H-1$ **do**
      $\hat{z}^{(n)}_{t+1} \leftarrow F_\theta(\hat{z}^{(n)}_t, a^{(n)}_t)$ ▷ World model predicted latent state
    end for
    Goal cost $c^{(n)}_g \leftarrow \left\|\hat{z}^{(n)}_H - z_g\right\|_2^2$
  end for
  Select the K sequences with lowest $c^{(n)}_g$ as elites $\mathcal{E}$: $\mu_j \leftarrow \frac{1}{K}\sum_{n\in\mathcal{E}} a^{(n)}_{0:H-1}$, $\Sigma_j \leftarrow \mathrm{Var}_{n\in\mathcal{E}}\left(a^{(n)}_{0:H-1}\right)$
end for
return optimized action sequence μ_J

Algorithm 3 Decision-time Planning with Cycle Action Consistency

Require: Encoder E_θ, world model F_θ, inverse dynamics G_ϕ, current observation o_0, goal observation o_g, planning horizon H, # of samples N, # of elites K, # of iterations J, consistency weight λ

Encode states: $z_0 \leftarrow E_\theta(o_0),\ z_g \leftarrow E_\theta(o_g)$
Initialize sampling distribution $\mathcal{N}(\mu_0, \Sigma_0)$ with $\mu_0 = \mathbf{0}$, $\Sigma_0 = I$
for $j = 1$ to J **do**
  Sample N action sequences $\{a^{(n)}_{0:H-1}\}_{n=1}^{N} \sim \mathcal{N}(\mu_{j-1}, \Sigma_{j-1})$
  for $n = 1$ to N **do**
    $\hat{z}^{(n)}_0 \leftarrow z_0$
    for $t = 0$ to $H-1$ **do**
      $\hat{z}^{(n)}_{t+1} \leftarrow F_\theta(\hat{z}^{(n)}_t, a^{(n)}_t)$ ▷ World model predicted latent state
      $\hat{a}^{(n)}_t \leftarrow G_\phi(\hat{z}^{(n)}_t, \hat{z}^{(n)}_{t+1})$ ▷ IDM inferred action
    end for
    Goal cost $c^{(n)}_g \leftarrow \left\|\hat{z}^{(n)}_H - z_g\right\|_2^2$
    Action consistency cost $c^{(n)}_a \leftarrow \frac{1}{H}\sum_{t=0}^{H-1}\left\|a^{(n)}_t - \hat{a}^{(n)}_t\right\|_2^2$
  end for
  Adaptive weight $w_a \leftarrow \lambda \cdot \sigma_g/\sigma_a$, where $\sigma_g \leftarrow \mathrm{std}_n\left(c^{(n)}_g\right),\ \sigma_a \leftarrow \mathrm{std}_n\left(c^{(n)}_a\right)$
  Augmented planning cost $c^{(n)} \leftarrow c^{(n)}_g + w_a\, c^{(n)}_a$ for $n = 1, \ldots, N$
  Select the K sequences with lowest $c^{(n)}$ as elites $\mathcal{E}$: $\mu_j \leftarrow \frac{1}{K}\sum_{n\in\mathcal{E}} a^{(n)}_{0:H-1}$, $\Sigma_j \leftarrow \mathrm{Var}_{n\in\mathcal{E}}\left(a^{(n)}_{0:H-1}\right)$
end for
return optimized action sequence μ_J

D.3 Planning solver

For each task, we apply the same planning configuration to all three world models. For the Cube, Reacher, and PushT, we followed the planning configuration of the original paper (Maes et al., 2026). We use a CEM solver that samples 300 action sequences per iteration and runs 30 optimization steps. In each step, the 30 candidates with the lowest cost are selected as elites to update the sampling distribution.

For the Rope and Granular tasks, we use a CEM solver that samples 200 action sequences per iteration for the Rope task and 100 action sequences for the Granular task, with both tasks running 10 optimization steps. In each step, the 30 candidates with the lowest cost are selected as elites to update the sampling distribution. Because the original DINO-WM paper (Zhou et al., 2025) did not release official planning settings for the Rope and Granular tasks, we selected the settings with which DINO-WM most closely matched the performance reported in that paper.

D.4 Decoders of LE-WM and PLDM, for visualization only

We train a lightweight transformer decoder that maps a frozen world-model CLS embedding back to pixels, used *for visualization only*. The CLS embedding serves as a single memory token attended

Table 6: **Planning performance on goal-conditioned visual navigation with NWM trained with CompACT.** The metrics are Absolute Trajectory Error (ATE) and Relative Pose Error (RPE), and lower is better for both. *Original* and *Ours* are as in Tab. 1, and parenthesized values denote the relative change (%).

Model	Planning cost	ATE	RPE (trans.)
NWM w/ CompACT	Original	1.3141	0.3831
	Ours	**1.2835 (-2.3%)**	**0.3773 (-1.5%)**

to by 196 learnable patch queries through a 4-layer, 8-head Transformer decoder (H=512), with each output token linearly mapped to a 16×16 RGB patch. Since the decoder sees only one vector, it cannot exploit encoder spatial features. We freeze each world model's encoder and projector and train the decoder with MSE on the corresponding expert dataset. Architecture, hyperparameters, and data are shared across Le-WM and PLDM, so qualitative differences reflect the representations, not the decoder.

E PLANNING WITH A VIDEO GENERATION-BASED WORLD MODEL

To test whether ACID extends beyond JEPA-style latent predictors, we apply it to NWM trained with CompACT (Bar et al., 2025; Kim et al., 2026), a world model based on video generation, on goal-conditioned visual navigation on RECON (Shah et al., 2021). As in the main experiments, we keep the world model and the CEM planner fixed and vary only the planning cost.

World model. NWM (Bar et al., 2025) uses a Conditional Diffusion Transformer to predict future observations from past observations and navigation actions, and we replace its SD-VAE tokenizer with CompACT-16 (Kim et al., 2026), which encodes each observation into only 16 discrete tokens to keep decision-time prediction tractable. We use the checkpoint released by the authors, trained with the 16-token version of the CompACT tokenizer, instead of reproducing the training.

Task and evaluation. Goal-conditioned visual navigation frames planning as an image-goal problem: given a current observation and a target image, the agent searches over action sequences by rolling out predicted future observations until it finds a path that reaches the goal. We evaluate on the RECON dataset (Shah et al., 2021), a large-scale collection of more than 5000 trajectories gathered autonomously in diverse real-world settings, with substantial visual variation driven by changing seasons and lighting conditions. Of the multiple sensing modalities available, we rely solely on the stereo RGB camera streams. We adopt the same experimental configuration as the original CompACT paper (Kim et al., 2026) and report two complementary metrics. Absolute Trajectory Error (ATE) captures the global fidelity of an estimated trajectory by measuring the Euclidean distance between matched points on the predicted and ground-truth paths, while Relative Pose Error (RPE) gauges local consistency by quantifying the discrepancy in the relative transformation between successive poses (Sturm et al., 2012).

Planning. The world model plans with a distance to the goal between observations (Zhang et al., 2018) as its goal cost, and we set the action consistency weight to $\lambda = 0.08$. Following the planning configuration of the original paper (Bar et al., 2025), we use a CEM solver that samples 80 action sequences per iteration and runs a single optimization step, selecting the 5 lowest-cost candidates as elites. The IDM integrates its learned velocity field with ten Euler steps.

Results. Tab. 6 shows that adding the action consistency cost lowers ATE by 2.3% and RPE by 1.5% relative to *Original*. The gain is smaller than on the manipulation tasks, but it is obtained without any change to the world model, indicating that the action consistency cost applies to a world model that generates future observations as well as to those that predict latent features.

Qualitative results. Since no environment rollout is available in this setting, Fig. 8 instead compares the ground-truth video against the world model's imagined rollout under the ground-truth actions: starting from the initial observation, we feed the ground-truth action sequence to the world model and compare its predicted frames against the ground-truth video. For each frame, we overlay the per-step action consistency score. The score stays low over the early frames, where the imagined rollout closely follows the ground-truth video, and rises toward the end, where the world model's rollout

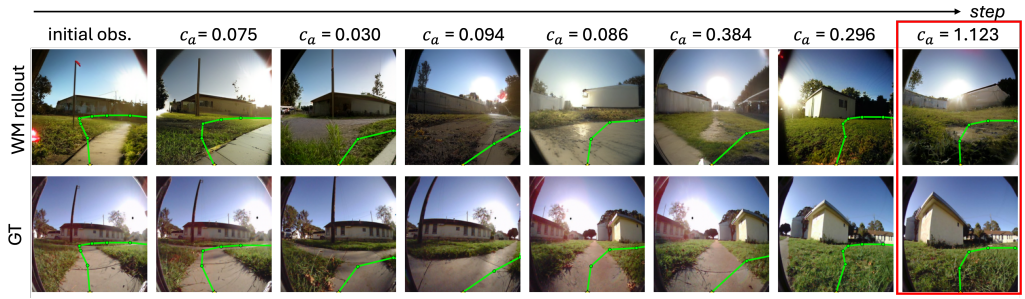

Figure 8: **Qualitative results on goal-conditioned visual navigation.** Comparison between ground-truth frames and world model rollouts. The action consistency score from ACID (*top*) and navigation trajectories (*green*) are presented together. Red box highlights the step where the action consistency cost drops; at this step, the predicted next frame fails to follow the action condition.



Figure 9: **Subgoal images of PickPlace and BowlStack.** From left to right: the pick and place goals of PickPlace, then those of BowlStack.

diverges from it. This shows that the action consistency score reliably identifies the unrealizable transitions, at which the world model fails to realize its conditioning action.

F Details of the real robot experiment

Robot and data. We use a PiPER arm with six joints and a gripper, observed by a single fixed RGB camera; the observation also includes the 7-dimensional joint state, and an action is an absolute joint target of the same dimension. Demonstrations are collected by teleoperation at 30 fps in the LeRobot format (Cadene et al., 2026). For PickPlace, we collect 200 episodes with the plate at a fixed position and the bowl at 57 distinct initial positions. For BowlStack, we collect 72 episodes, placing one bowl at a distinct initial position in each episode while keeping the other bowl at a fixed position. For each task, the world model and the IDM are trained on all of its episodes, and the diffusion policy is trained on a 30-episode subset covering 30 bowl positions for PickPlace and on all 72 episodes for BowlStack. Each task is specified by two subgoal images, a pick goal and a place goal, as shown in Fig. 9.

Proposal policy. The policy is a 1D-UNet Diffusion Policy (Chi et al., 2023) with a ResNet-18 image encoder, conditioned on the two most recent camera frames and joint states. It predicts a 16-step action chunk at 10 Hz, of which the first 15 steps are executed. At deployment, each of the N candidates is sampled with a deterministic DDIM sampler from its own initial noise, so the diversity among the candidates comes only from the initial noise. The hyperparameters are listed in Tab. 7.

Table 7: **Hyperparameters of Diffusion Policy.**

Name	Value
Observation horizon	2
Prediction / execution horizon	16 / 15
DDIM steps (training / deployment)	100 / 10
Optimizer	AdamW
Learning rate	10^{-4}
Batch size	64
EMA	Yes

Table 8: **Dataset details and world model training configurations for the real robot experiment.** Frames are recorded at 30 fps. History length, frame skip, and train epochs are for training the world model.

Task	Size	# Frames	History len	Frame skip	Train epoch
PickPlace	200	325∼638	3	45	10
BowlStack	72	258∼395	3	15	10

World model and inverse dynamics verifier. The world model follows the DINO-WM architecture (Zhou et al., 2025), with the frozen DINOv2 encoder replaced by a ResNet encoder trained jointly with the predictor under the temporal straightening regularizer of Wang et al. (2026) with weight 0.1. The encoder maps each frame to a 16×16 grid of 8-dimensional tokens, and the joint state and the action chunk are each embedded to 10 dimensions and concatenated to every visual token. The predictor is a causal ViT with 6 layers, 16 heads, and MLP width 2048 that takes three past frames and predicts the next latent. It is trained with AdamW, with a learning rate of 5×10^{-4} for the predictor and 10^{-5} for the encoder, and a batch size of 32. The dataset details and training configurations are listed in Tab. 8. The IDM is the flow-matching action decoder of Appendix B with 4 transformer layers and width 192, trained on the frozen latents of this world model; at planning time it recovers the action chunk with a single Euler step.

G Comparison of CEM and ACID.

We provide the pseudo-code for each test-time optimization method in Algorithm 2 and Algorithm 3. Algorithm 3 is identical in content to the version presented in the main paper, with the components that differ from Algorithm 2 highlighted in a distinct text color.

H Empirical justification for the scale-invariant adaptive weight

In the main section Sec. 3.3 we set the action consistency weight adaptively as $w_a = \lambda \cdot \sigma_g / \sigma_a$ (Eq. 7 of the main), motivated by the claim that the relative spread of the goal cost c_g and the action consistency cost c_a varies across (1) world models, (2) tasks, and (3) CEM iterations. Here we provide the empirical measurements behind that claim (Fig. 10).

Measurement protocol. We define a pool as the set of N CEM candidates evaluated for a single environment at a single (MPC step, CEM iteration). Within each pool we compute σ_g, the standard deviation of the goal cost across the N candidates, and σ_a, the standard deviation of the action consistency cost. Because many pools exist at each CEM iteration (one per MPC step per environment), each plotted point reports the *median* of σ_g, σ_a, or their ratio over all pools at that iteration; we use the median rather than the mean for robustness to outlier pools. The number of candidates N and CEM iterations differs per world model (see Sec. D.3 for the full settings); the standard deviation is computed within each pool and is therefore unaffected by these differences in pool count. The y-axis is log-scaled in all panels, and the gray dashed reference at $\sigma_g / \sigma_a = 1$ in panel (c) marks the regime where the two costs carry equal within-pool spread.

Why spread, not magnitude, governs reranking. CEM ranks candidates by the augmented cost $c_g + w_a c_a$ and keeps the lowest-cost elites, so c_g and c_a jointly determine which candidates become elites. A constant offset added to every candidate's cost leaves the ranking unchanged; only *differences* across candidates move the ordering. Hence the influence of each term on elite selection is set by how widely it varies across the pool—its standard deviation—not by its absolute level. This is why we track σ_g and σ_a rather than the raw cost values.

The relative spread varies along all three axes. Figure 10 confirms the three sources of variation, with one row per world model (Le-WM, PLDM, DINO-WM) and one curve per task. (1) *Across world models.* The absolute scale of σ_g in column (a) differs by orders of magnitude across rows, and the ratio σ_g / σ_a in column (c) sits at a different overall level in each row. Different latent spaces thus induce inherently different cost distributions. (2) *Across tasks.* Within a single row—i.e., fixing the world model—the values of σ_g and σ_a still differ across tasks, so the two spreads, and their

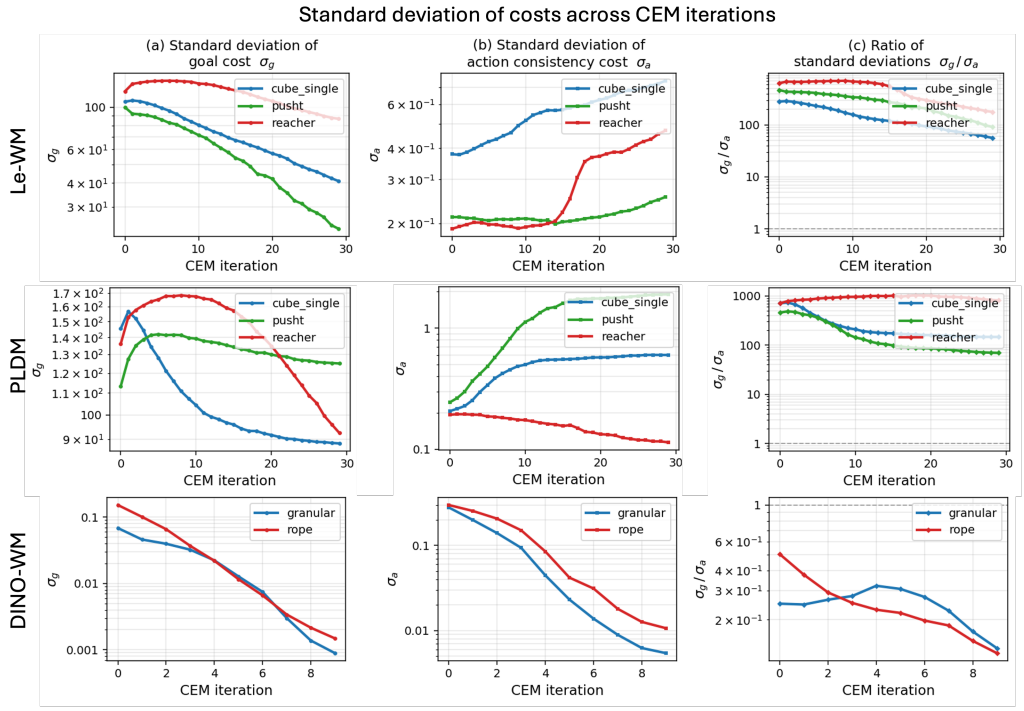

Figure 10: **Within-pool spread of the goal cost and action consistency cost.** Rows are world models; columns show the within-pool standard deviations σ_g (a) and σ_a (b) and their ratio σ_g/σ_a (c) on a log-scaled y-axis, with one curve per task. Each point is the median over all pools at that CEM iteration. The dashed line at $\sigma_g/\sigma_a = 1$ marks equal within-pool spread.

ratio in column (c), take task-dependent values. (3) *Across CEM iterations.* Both σ_g and σ_a change continuously over the course of a single optimization run, and the size of the resulting drift in the ratio is itself world-model-dependent: in Le-WM and PLDM the two spreads compress at different rates and σ_g/σ_a in column (c) shifts by up to an order of magnitude, whereas in DINO-WM they decay at nearly matching rates and the ratio drifts more mildly. In every case, however, the ratio is far from constant across world models and tasks, so a weight calibrated on one configuration does not transfer to another.

Why the adaptive weight works. Because the ratio is far from constant across all three axes, no single fixed w_a can keep the consistency term's contribution $w_a\sigma_a$ at a representative level: it overrides the goal cost where σ_a is large and vanishes where σ_a is small. The adaptive weight removes this dependence by recomputing σ_g/σ_a at every iteration, so the weighted spread $w_a\sigma_a = \lambda\sigma_g$ tracks the goal-cost spread in every pool. This leaves only λ as a free parameter, which encodes how strongly to prioritize the consistency cost relative to the goal cost. We find empirically that a single λ per world model transfers across tasks without per-task retuning (see Sec. 4.4 of the main for the supporting experiment).

I Per-iteration overhead of inverse dynamics verifier

Table 9: **Per-iteration latency of world models and IDM verifiers.** t_{WM} is the wall-clock time of the world model rollout and t_{IDM} that of the inverse dynamics verifier forward within one CEM iteration.

World model	t_{WM} (ms)	t_{IDM} (ms)	$t_{\mathrm{IDM}}/t_{\mathrm{WM}}$ (%)
Le-WM	17.54	2.54	14.5
PLDM	17.33	2.51	14.5
DINO-WM	2,080.50	245.25	11.8

We report the wall-clock cost of the two networks that evaluate the planning cost within one CEM iteration: the world model rollout and the inverse dynamics verifier. Each is timed over one CEM iteration with 300 candidate action sequences and a planning horizon of 5. We report the verifier overhead as the ratio of the two times. All measurements are taken on a single RTX 6000 Ada GPU.

J RANKING CANDIDATES WITH THE ACTION CONSISTENCY COST

Table 10: **Ranking candidates with the action consistency cost.** Each of the 19 distractors is the ground-truth action chunk plus Gaussian noise, with a standard deviation equal to that of the actions in the dataset. *Original* ranks the candidates with the goal cost only, while *Ours* uses the augmented cost of Equation 6.

Task	Model	Recall@1 (%)			Spearman ρ	
		Original	Ours	Δ	Original	Ours
PushT	PLDM	70.3	91.0	+20.7	0.44	0.51
	DINO-WM	67.5	89.2	+21.7	0.42	0.48
	Le-WM	68.1	89.8	+21.7	0.44	0.51
Reacher	PLDM	32.1	49.9	+17.8	0.38	0.43
	DINO-WM	22.8	34.8	+12.0	0.33	0.37
	Le-WM	34.2	54.6	+20.4	0.39	0.43
Cube	PLDM	75.3	95.9	+20.6	0.35	0.40
	DINO-WM	79.5	95.9	+16.4	0.40	0.43
	Le-WM	91.1	98.4	+7.3	0.35	0.39

Inside the planner, the planning cost scores candidates: the goal cost c_g asks whether a candidate's predicted outcome is close to the goal, while the action consistency cost c_a asks whether that outcome was actually produced by the candidate's actions. We test whether c_a separates candidates that c_g cannot.

Setup. On the validation set of each task, we rank 20 candidate chunks, the ground-truth chunk and 19 distractors, using the planner's cost of Equation 6 with $\lambda = 0.07$. Each distractor is obtained by adding Gaussian noise to the ground-truth chunk, with a standard deviation equal to that of the actions in the dataset. We report the recall@1 of the ground-truth chunk in percent and the Spearman ρ between the cost ranking and the action-space distance ranking. A random ranking yields a recall@1 of 5% and a ρ of 0.

Results. As shown in Tab. 10, adding c_a raises both recall@1 and ρ for every world model on every task. Where c_g cannot separate the candidates, c_a restores the ranking by detecting that the action recovered from each prediction disagrees with the candidate that conditioned it. The IDM thus serves as the verifier the planner requires, consistently across world models and tasks.

K QUALITATIVE RESULTS

In Fig. 11, we qualitatively compare planning with and without ACID, for each baseline world model and task. To enable visual inspection, we train an auxiliary decoder that maps latent states back to pixels, letting us check whether the imagined rollout in latent space agrees with the rollout obtained in the actual environment. In each figure, the top row shows the pixel-decoded predictions of the world model (the imagined rollout under the planned action sequence), and the bottom row shows the environment rollout obtained by executing that same action sequence. Without ACID, the imagined rollout often arises from unrealizable transitions: it reaches a goal-like state in the world model's prediction, but executing the corresponding action sequence drives the environment elsewhere. With ACID, the planned action sequence is realizable—the environment rollout closely tracks the imagined rollout, confirming that the predicted goal-reaching can actually be reproduced.

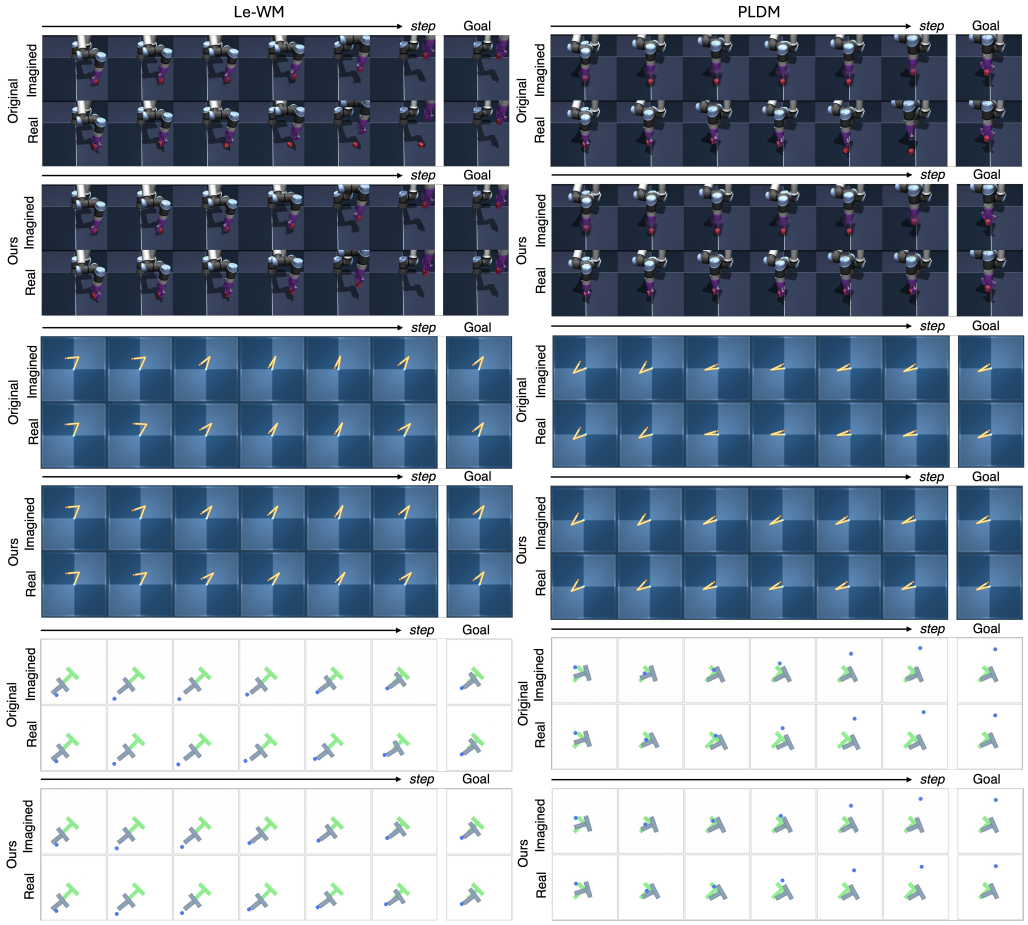

Figure 11: **Qualitative comparison on Le-WM and PLDM across the OGBench-Cube, Reacher, and Push-T tasks.** Le-WM is shown in the left column and PLDM in the right column. For each model and task, we show the real rollout in the environment and the corresponding imagined rollout from the world model under the planned action sequence, comparing the baseline with the baseline with ACID.