ERRATAWM: CORRECTING FROZEN WORLD MODELS WITH A NON-PARAMETRIC MEMORY OF THEIR OWN ERRORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent world models enable strong planning, but once trained they are static: deployed in new environments with shifted dynamics they keep making the same systematic prediction errors, and those errors flow straight into the planner. The usual remedy, gradient-based test-time training, is expensive, rewrites the whole model, and tends to forget the environments it was already good at. We first establish an empirical premise: a frozen world model's errors are not noise but a function of the starting context, so that similar states yield similar mistakes. This holds broadly, across navigation, manipulation, and driving, five world models, and four architectures including regression, autoregressive, and video diffusion models, with context-specific predictability as high as 0.81. Building on this, we propose ERRATAWM, a frozen world model paired with a non-parametric memory that turns its recurring errors into scope-gated correction prototypes and applies them at test time with no gradient updates. Because the memory operates in a shared frozen feature space, one model corrects heterogeneous world models across tasks. ERRATAWM reduces one-step prediction error by a context-specific margin of 0.12 to 0.15 over a context-blind control, matches gradient-based adaptation without any weight update, and improves downstream success by 9 to 13 points on navigation and manipulation. A single shared memory corrects a navigation and a manipulation world model at once and transfers zero-shot to an unseen driving diffusion model, reaching success competitive with recent test-time world-model methods at a fraction of their adaptation cost[1].

1 INTRODUCTION

World models have become a central tool for decision making: by predicting how an environment will evolve, an agent can plan its actions by imagination rather than by trial and error (Ha & Schmidhuber, 2018; Hafner et al., 2019; Schrittwieser et al., 2020; Hafner et al., 2025). Recent models make this practical at scale. Latent and feature-space predictors forecast in the representation of a frozen visual encoder and enable zero-shot planning (Zhou et al., 2025; Assran et al., 2025), while large generative video models act as general world simulators across navigation, manipulation, and driving (Bar et al., 2025; Gao et al., 2024; NVIDIA et al., 2026). In all of these, planning quality is bounded by prediction quality: a controller that rolls out an inaccurate model inherits its mistakes.

The difficulty is that these models are static once trained. Deployed in an environment whose dynamics differ from training, a frozen world model does not fail randomly; it makes the same systematic mistakes whenever it meets the same kind of situation, and every such error propagates directly into the plan built on top of it. The natural fix is to keep learning at test time, and gradient-based test-time training does exactly this (Sun et al., 2020; Wang et al., 2021), including recent methods that adapt a world model or policy online (Wan et al., 2026; Zhou et al., 2026; Jang et al., 2026; Huang et al., 2025; An et al., 2026). But updating the weights is costly, it rewrites the entire model, and it tends to forget the environments the model was already good at, which is a poor fit for an agent that must remain reliable everywhere it has been.

[1]The code and checkpoints will be released upon acceptance.

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

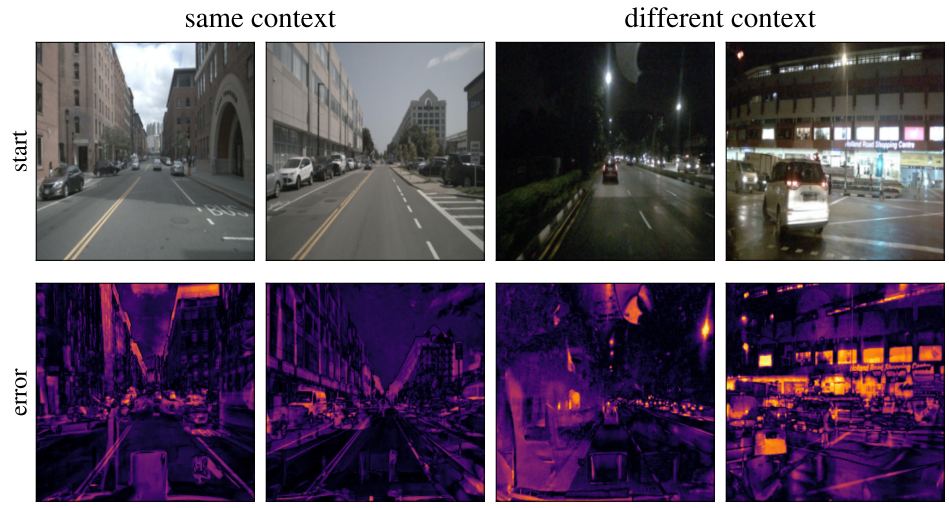


Figure 1: A frozen world model repeats its mistakes. Two different scenes of the same context (left) yield nearly identical error heatmaps, while scenes from other contexts (right) fail differently ($+0.77$ within a context vs. $+0.39$ across; Cosmos-Predict2 (NVIDIA et al., 2026) on nuScenes (Caesar et al., 2020)). ERRATAWM remembers these recurring errors and cancels them at test time.

We start from a different observation. If a frozen model's mistakes recur with the situation, then its error is predictable from the situation, and a mistake seen once can be anticipated and cancelled the next time a similar situation arises, without changing a single weight. Figure 1 shows this directly: a frozen model produces nearly the same error pattern on two different scenes of the same kind, yet different patterns on scenes of other kinds, so its mistakes are structure a memory can capture and cancel. We test this premise directly and find it holds broadly: the error of a frozen world model is well predicted by the starting context across navigation, manipulation, and driving, across five world models, and across four architectures spanning regression, autoregressive, and pixel and video diffusion models, with context-specific predictability as high as 0.81 even for large pretrained models. Errors are structure, not noise.

Building on this, we propose ERRATAWM, a frozen world model augmented with a non-parametric memory of its own prediction errors. As the agent acts, ERRATAWM records recurring errors as correction prototypes, each pairing the context in which the model errs with a compact correction and a learned scope describing where that correction applies and, from negative evidence, where it must stay silent. At test time the current context queries the memory, retrieves the applicable corrections, and adds them to the frozen prediction, so adaptation happens by retrieval rather than by gradient descent. Because the memory lives in the shared frozen feature space, it is agnostic to the world model that produced the prediction: feature-native models are read directly and pixel or video models are re-encoded, so one memory can correct heterogeneous world models across tasks. The memory grows with new environments instead of overwriting old ones, enabling ERRATAWM to adapt without forgetting.

Empirically, ERRATAWM corrects a frozen world model's one-step prediction by a context-specific margin of 0.12 to 0.15 over a context-blind control, and matches gradient-based test-time training while performing no weight update. These gains carry into control, improving downstream success by 9 to 13 points on navigation and manipulation, and a single shared memory corrects a navigation and a manipulation world model at once while transferring zero-shot to an unseen driving diffusion model. Against recent test-time world-model methods, ERRATAWM reaches competitive success at a fraction of the adaptation cost and with no forgetting, and ablations show that a simple contrastive control objective is what makes its corrections context-specific rather than a global bias.

Our **contributions** are: **(i)** an empirical premise, that a frozen world model's errors are predictable from context, established across three tasks, five world models, and four architectures; **(ii)** ERRATAWM, a gradient-free, non-parametric error memory that corrects frozen world models at test time in a shared feature space, making it agnostic to the underlying world model; and **(iii)** experiments

showing that ERRATAWM corrects prediction and improves control across navigation, manipulation, and driving, that one memory serves heterogeneous world models and transfers to an unseen one, and that it is competitive with gradient-based test-time methods at far lower cost.

2 RELATED WORK

Memory- and retrieval-augmented world models. Recent world models increasingly augment parametric predictions with external memory, letting past experience or retrieved knowledge restore missing context and improve long-horizon consistency. In robotic manipulation, Ctrl-World (Guo et al., 2026) conditions prediction on sparse historical observations and pose information, while Mem-World (Zheng et al., 2026) explicitly retrieves geometrically relevant past views to preserve objects across occlusion and camera motion. R-WoM (Mei et al., 2026) similarly augments an LLM-based world model with retrieved environment-specific procedural knowledge for computer-use agents. These approaches build on a broader line of non-parametric prediction and episodic memory, where similar past contexts are retrieved at inference time to complement a parametric model (Khandelwal et al., 2020; Pritzel et al., 2017). Existing memories, however, primarily store observations, states, trajectories, or external knowledge about the environment. In contrast, ERRATAWM treats the world model's own failures as reusable experience: it stores context-indexed prediction errors and retrieves corrections, rather than content, when similar situations recur.

Test-time adaptation of world models and policies. A growing line of work adapts models during deployment to cope with distribution shift, from generic test-time adaptation that updates representations via self-supervised or entropy objectives (Sun et al., 2020; Wang et al., 2021) to world-model-specific methods: WorldAgen (Wan et al., 2026) fine-tunes a shared world-model and policy representation from target transitions, AdaPower (Huang et al., 2025) performs parameter-efficient test-time training on a mostly-frozen foundation model, and TMoW (Jang et al., 2026) adapts how multiple world models are routed and combined. Closest to us, Feedback-WM (An et al., 2026) feeds the discrepancy between predicted and realized latents back into subsequent predictions through an online observer, and τ_0-WM (Zhou et al., 2026) rescores and rectifies candidate actions with extra test-time computation. ERRATAWM differs in that it performs no gradient update or expert reconfiguration, adapting only by writing to and querying a persistent non-parametric memory; unlike τ_0-WM it pays only a single retrieval per step and keeps its corrections across interactions, and unlike Feedback-WM's transient observer state it stores context-indexed error prototypes that can be reused when the same failure reappears much later.

Local and residual dynamics correction. The idea of correcting a nominal dynamics model with a locally estimated residual is classical in control. Locally weighted and model-on-demand methods construct data-driven dynamics models around the current operating point and use them for prediction and model-predictive control (Atkeson et al., 1997; Braun et al., 2000), while more recent methods explicitly regress model-prediction errors for MPC (Xue et al., 2024) or learn residual dynamics to adapt an offline model under environment shift (Nakhaei et al., 2024). The basic decomposition $f_{\text{true}}(x, a) \approx f_{\text{nominal}}(x, a) + \Delta f(x, a)$ is therefore not itself new. ERRATAWM instead asks how this principle can be made practical for modern, high-dimensional foundation world models. Rather than defining locality through low-dimensional state distances or hand-designed neighborhoods, it retrieves corrections through a learned semantic context representation; its scope mechanism learns when retrieved experience is applicable and when the base prediction should be left unchanged; and prediction and observation are mapped into a shared feature space, so the same error-memory interface operates across heterogeneous world models, including latent, pixel or video, and autoregressive predictors.

3 THE PREMISE: WORLD-MODEL ERRORS ARE CONTEXT-PREDICTABLE

ERRATAWM rests on a simple empirical claim: a frozen world model does not err randomly, it makes similar mistakes from similar starting states. If true, its error is a function of context and can be predicted, and therefore corrected, from past experience. We test this directly across three tasks, several world models, and four architectures, before building any memory.

Table 1: Context-specific error predictability Δ (Eq. (2)); higher is more predictable, and the shuffled-context control is already subtracted, so $\Delta > 0$ means the error depends on context. The premise holds across tasks, world models, and architectures.

Task	World model	Architecture	Dataset	Δ
Navigation	NWM (Bar et al., 2025)	diffusion	RECON	$+0.81$
Manipulation	DINO-WM (Zhou et al., 2025)	regression	LIBERO	$+0.70$
Manipulation	DINO-WM (Zhou et al., 2025)	regression	CALVIN	$+0.43$
Manipulation	Cosmos (NVIDIA et al., 2026)	diffusion	LIBERO	$+0.56$
Manipulation	iVideoGPT (Wu et al., 2024)	autoregressive	LIBERO	$+0.21$
Manipulation	iVideoGPT (Wu et al., 2024)	autoregressive	CALVIN	$+0.25$
Driving	Vista (Gao et al., 2024)	video diffusion	nuScenes	$+0.49$
Driving	Cosmos (NVIDIA et al., 2026)	video diffusion	nuScenes	$+0.45$

For each transition we form a context $c = [\mathrm{pool}(z_t) \| a_t]$ and the residual $r = z_{t+1} - \hat{z}_{t+1}$ in the frozen feature space of Eq. (3), and we ask how well r is predicted by the residuals of its context-neighbors. Given a source pool, we estimate a query residual by its k nearest neighbors in context space, $\hat{r} = \frac{1}{k} \sum_{j \in \mathcal{N}_k(c)} r_j$, and measure the relative error reduction

$$G \;=\; 1 - \frac{\mathbb{E}\,\|r - \hat{r}\|_2^2}{\mathbb{E}\,\|r\|_2^2}. \tag{1}$$

A large part of G can come from a global bias shared by all predictions, so we subtract a control G_{shuf} computed with neighbors drawn from shuffled contexts, and report the context-specific predictability

$$\Delta \;=\; G - G_{\mathrm{shuf}}. \tag{2}$$

A value $\Delta > 0$ means the error carries structure that depends on the starting context beyond any global offset, which is exactly the signal a context-keyed memory can exploit.

Table 1 reports Δ across settings, and it is positive everywhere: navigation, manipulation, and driving; feature-native regression models, autoregressive models, and pixel and video diffusion models; and both in-distribution and out-of-distribution data. Even large pretrained models are highly predictable, with NWM on RECON reaching $\Delta = 0.81$, and a general video foundation model with no driving fine-tuning still yields $\Delta = 0.45$ on nuScenes while its shuffled control sits at -0.14, confirming the signal is genuinely context-specific rather than a global bias. Because this holds regardless of task, architecture, or whether the model is feature-native or generative, a single non-parametric memory keyed on context can recover and reuse these corrections, which motivates ERRATAWM (§4).

4 ERRATAWM

We augment a frozen world model with a non-parametric memory that corrects its recurring prediction errors at test time, without any gradient update to the world model (Fig. 2). Let f be a frozen encoder that maps an observation o_t to patch features $z_t = f(o_t) \in \mathbb{R}^{N \times d}$ (Oquab et al., 2024), and let the base world model predict the next state from a short history and action a_t, read into the same feature space as $\hat{z}_{t+1}$ (Dawid & LeCun, 2024; Zhou et al., 2025). The quantity we care about is the model's error, the residual

$$r_t \;=\; z_{t+1} - \hat{z}_{t+1}. \tag{3}$$

Our premise (§3) is that r_t depends on the starting context rather than being noise: from similar states the model makes similar mistakes. ERRATAWM exploits this with a memory of context-indexed corrections, retrieved by similarity and applied additively in feature space, and then used for goal-conditioned latent model-predictive control.

4.1 ERRATA MEMORY

The memory is a bounded set of prototypes, each summarizing a recurring class of error rather than a single experience:

$$\mathcal{M} = \{m_k\}_{k=1}^{K}, \qquad m_k = \big(\mu_k,\, S_k,\, \mu_k^-,\, S_k^-,\, \delta_k,\, n_k,\, \omega_k\big). \tag{4}$$

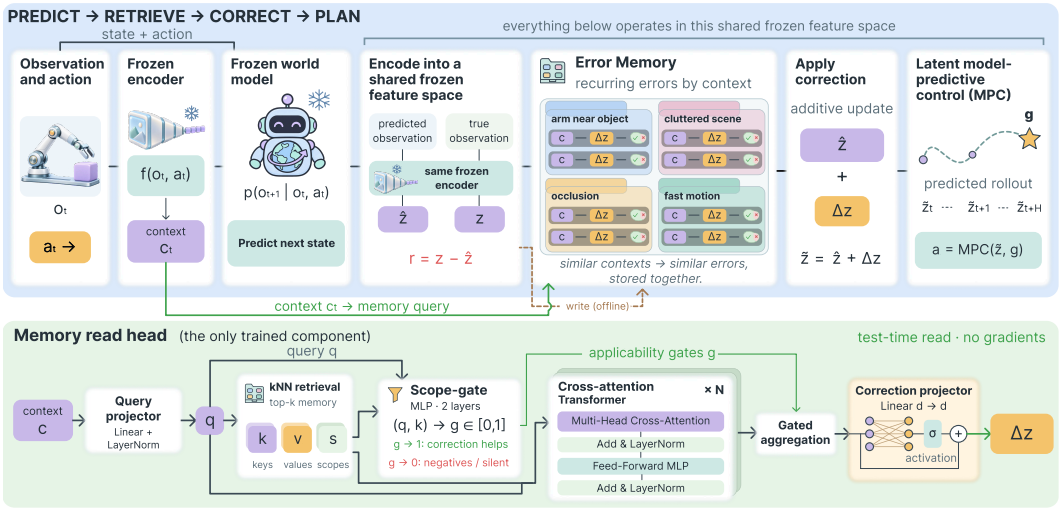


Figure 2: **Overview of ErrataWM.** A frozen world model predicts the next state; its prediction and the true observation are encoded by the same frozen encoder into a shared feature space, where the error $r = z - \hat{z}$ is defined. An error memory stores recurring errors as prototypes (a context key, a correction Δz, and a learned scope), grouped by context. At test time the context c_t queries a learned read head, the only trained module, which retrieves and scope-gates prototypes into a correction Δz; the corrected latent $\tilde{z} = \hat{z} + \Delta z$ drives model-predictive control. The world model and encoder stay frozen and adaptation is gradient-free.

Here μ_k, S_k are the center and diagonal scope of the contexts where the correction applies, and μ_k^-, S_k^- are the same for contexts where it must stay silent. The vector δ_k is a compact correction code, n_k a support count, and $\omega_k \in [0, 1]$ a running reliability. This mirrors the classical residual decomposition, a global nominal model plus a local learned correction (Atkeson et al., 1997; Xue et al., 2024; Nakhaei et al., 2024), lifted to a high-dimensional world model with learned semantic retrieval (Khandelwal et al., 2020; Pritzel et al., 2017).

Three small trained modules connect predictions to memory. A context encoder describes the current situation, and a residual encoder turns an observed error into a correction code:

$$c_t = \phi_{\text{ctx}}\big(\text{pool}(z_t),\, a_t\big) \in \mathbb{R}^{d_k}, \qquad u_t = \phi_{\text{res}}\big(r_t,\, z_t\big) \in \mathbb{R}^{d_v}. \tag{5}$$

A FiLM applier $U_\xi(\hat{z}_{t+1}, \delta_k)$ then turns a stored code into a spatial correction on the current prediction. With f a frozen DINOv2 ViT-B/14 (N=256 tokens, d=768), each module is a two-layer MLP over attention-pooled features with d_k=d_v=256, about one million parameters each. The action is embedded by a per-dimension tokenizer that accepts a variable action dimension up to seven, so one model spans navigation (3-DoF) and manipulation (7-DoF). The applier's output projection is zero-initialized, so an untrained memory is exactly the identity.

To decide which prototypes apply, we score the context by positive and negative Mahalanobis distances under the stored scopes:

$$D_k^{\pm}(c_t) = \sum_j \frac{\big(c_{t,j} - \mu_{k,j}^{\pm}\big)^2}{S_{k,j}^{\pm} + \varepsilon}. \tag{6}$$

Each prototype then has an independent gate with only four trainable scalars,

$$g_{t,k} = \sigma\big(-\eta^+ D_k^+(c_t) + \eta^- D_k^-(c_t) + \kappa\,\text{logit}(\omega_k) + b\big), \tag{7}$$

so a prototype fires when the context is near its positive scope, far from its negative scope, and reliable. The correction is a bounded, gated sum over prototypes:

$$\hat{z}_{t+1}^M = \hat{z}_{t+1} + \sum_k w_{t,k}\, U_\xi\big(\hat{z}_{t+1}, \delta_k\big), \qquad w_{t,k} = \frac{g_{t,k}}{1 + \sum_j g_{t,j}}. \tag{8}$$

When no gate fires the memory is silent, and the normalization keeps the total correction bounded even when many gates fire. Reading uses only retrieval and forward passes, unlike gradient-based test-time training (Sun et al., 2020; Wang et al., 2021) or online-observer correction (An et al., 2026).

4.2 MEMORY DYNAMICS

After observing r_t, the memory is updated by closed-form online rules with no gradient step. An event is assimilated into its nearest prototype by a reliability-weighted running update of the center, scope, and code, and a new prototype is created when none is close enough. Reliability is updated only when a prototype is actually applied, using discounted hit and miss counts that penalize harmful applications more than they reward helpful ones and let stale prototypes lose trust. When an applied prototype increases the error, its context is recorded as negative evidence in (μ_k^-, S_k^-). The positive scope therefore learns where a correction applies, while reliability and the negative scope learn where it must stay silent. Overgrown prototypes are split and near-duplicates merged, and the store is capped at $K_{\max}$ (128 in training, up to 1024 at deployment), so it remains a few megabytes with bounded per-step cost.

4.3 TRAINING

The encoder and base world model are frozen; we train only the two encoders, the applier, and the four gate scalars. Training is episodic: each episode is split into disjoint support and query sets, the memory is built from the support set alone, and it is evaluated on the query set, so any measured gain reflects generalization rather than memorization. Writing z^0 for the uncorrected prediction and z^M for the corrected one, the objective combines four terms,

$$\mathcal{L}_{\text{mem}} = \|z^M - z\|_2^2, \tag{9}$$

$$\mathcal{L}_{\text{gain}} = \big[\, \|z^M - z\|_2^2 - \|z^0 - z\|_2^2 + \delta_m \,\big]_+, \tag{10}$$

$$\mathcal{L}_{\text{scope}} = \sum_{k \in \mathcal{K}_t} \text{BCE}\big(g_{t,k},\, y_{t,k}\big), \tag{11}$$

$$\mathcal{L}_{\text{control}} = \big[\, \|z^M - z\|_2^2 - \|z^{M,\text{shuf}} - z\|_2^2 + m \,\big]_+, \tag{12}$$

combined as

$$\mathcal{L} = \mathcal{L}_{\text{mem}} + \lambda_g\, \mathcal{L}_{\text{gain}} + \lambda_s\, \mathcal{L}_{\text{scope}} + \lambda_c\, \mathcal{L}_{\text{control}}. \tag{13}$$

Each term plays a distinct role. $\mathcal{L}_{\text{mem}}$ drives the corrected prediction toward the truth. $\mathcal{L}_{\text{gain}}$ is a hinge that credits only corrections which beat the no-memory baseline. $\mathcal{L}_{\text{scope}}$ supervises the gate against counterfactual usefulness labels $y_{t,k}$ on the near-context candidate set $\mathcal{K}_t$. Finally $\mathcal{L}_{\text{control}}$ compares against $z^{M,\text{shuf}}$, a read whose prototypes are assigned from a shuffled context, forcing the memory to beat a context-agnostic global shift so it captures genuinely context-specific error. We keep the discrete bookkeeping (assignment, counts, scopes, births, and merges) detached and let gradients flow only through the accumulated codes, so the writes used in training match the rule-based, gradient-free writes at deployment.

4.4 WRAPPING HETEROGENEOUS WORLD MODELS

Because everything after Eq. (3) lives in f's feature space, a single memory wraps different world models. Feature-native models predict there directly (Zhou et al., 2025; Assran et al., 2025), while pixel and video models (Bar et al., 2025; Gao et al., 2024; Wu et al., 2024; NVIDIA et al., 2026) generate a next observation $\hat{o}_{t+1}$ that we re-encode as $\hat{z}_{t+1} = f(\hat{o}_{t+1})$. For a diffusion base we take a deterministic fixed-noise readout so the residual reflects systematic error rather than sampling noise. We instantiate navigation with a frozen NWM, manipulation with a DINO-WM we train, and the diffusion case with a frozen Cosmos-Predict2 video model, all corrected by a single memory.

5 EXPERIMENTS AND RESULTS

5.1 EXPERIMENTAL SETUP

Tasks. We evaluate on three tasks that stress different dynamics: visual navigation, robotic manipulation, and autonomous driving. Navigation and manipulation are the settings where we train and deploy the memory, since both provide action-conditioned world models and held-out environments with genuine dynamics shift. Driving is used as a third domain for the premise study (§3) and, through

Table 2: One-step prediction correction (relative latent-error reduction ↑; Hurt is the fraction of steps a correction worsens the prediction ↓). Best correction per row in bold. Standard deviations over three seeds are reported in Table 10 (Appendix B).

Task	Benchmark	World model	Floor	Rand-ctx	kNN	TENT	TTT	ERRATAWM	Hurt
Nav	RECON (in)	NWM	0.000	0.043	0.112	0.096	0.127	**0.168**	0.184
Nav	RECON (OOD)	NWM	0.000	0.076	0.183	0.214	0.196	**0.224**	0.267
Manip	LIBERO	DINO-WM	0.000	0.052	0.137	0.118	0.149	**0.191**	0.216
Manip	CALVIN	DINO-WM	0.000	0.037	0.104	0.089	0.123	**0.157**	0.173
Manip	LIBERO	Cosmos	0.000	0.064	0.158	0.174	0.162	**0.203**	0.238
Manip	LIBERO	iVideoGPT	0.000	0.029	0.091	0.073	0.106	**0.139**	0.147
Drive	nuScenes	Vista	0.000	0.071	0.169	0.193	0.181	**0.216**	0.251
Drive	nuScenes	Cosmos	0.000	0.078	0.194	**0.228**	0.207	0.227	0.283

the feature-space interface of §4, as a test of correcting a general video foundation model that never saw driving during fine-tuning.

World models. All world models are frozen. For navigation we use NWM (CDiT-S/2) (Bar et al., 2025); for manipulation we train a DINO-WM feature predictor (Zhou et al., 2025) on the target suite; and for the generative and foundation settings we use the pixel and video models iVideoGPT (Wu et al., 2024), Vista (Gao et al., 2024), and Cosmos-Predict2 (NVIDIA et al., 2026), together with the feature-native V-JEPA 2 (Assran et al., 2025). Feature-native models are read directly; pixel and video models are re-encoded with the frozen DINOv2 ViT-B/14 encoder (Oquab et al., 2024) before the residual of Eq. (3), and diffusion models use a deterministic fixed-noise readout.

Baselines. We compare ERRATAWM against four alternatives sharing the same frozen world model. The no-memory floor is the world model alone, the reference every gain is measured against. The random-context control retrieves corrections from shuffled contexts, isolating context-specific gain from any global bias. The kNN baseline applies raw nearest-neighbor residual corrections without the learned scope-gated read, testing the value of the read head over plain retrieval (Khandelwal et al., 2020; Pritzel et al., 2017). Finally, gradient-based test-time adaptation fine-tunes a lightweight adapter on the deployment stream (Sun et al., 2020; Wang et al., 2021), the parametric counterpart to our non-parametric memory, which we compare against for both accuracy and stability under continued shift. Refer to Appendix A for details on the baselines.

Benchmarks. Navigation uses RECON (Shah et al., 2023), manipulation uses LIBERO (Liu et al., 2023) and CALVIN (Mees et al., 2022), and driving uses nuScenes (Caesar et al., 2020). For every benchmark we build disjoint meta-training and deployment splits so that the environments a prototype is evaluated on are never those it was written from, and the residual statistics used for whitening are always taken from the meta-training split, never the deployment split. Refer to Appendix A for details on the benchmarks.

Training details. The DINOv2 ViT-B/14 encoder is frozen at 224×224. The DINO-WM predictor is a six-layer, eight-head transformer with hidden width 512 and a three-frame history, trained on features at a 16-step horizon. The memory modules are trained with AdamW at learning rate 3×10^{-4}, weight decay 0.01, a 500-step warmup, and gradient clipping at 1.0, for 6000 episodic steps with 8 episodes per step over three seeds. Each episode draws 24 to 96 support and 96 query transitions, and the store is capped at 128 prototypes during training and 1024 at deployment (with $k = 4$ neighbors and a single applied prototype per step). For the objective of Eq. (13) we use $\lambda_g = 1.0$, $\lambda_s = 0.5$, $\lambda_c = 1.0$ and set both hinge margins to 0. The foundation memory is trained on navigation and manipulation, with domain-specific residual whitening. Appendix F reports sensitivity to the number of applied prototypes, the store budget, and the loss weights; the method is robust across these, and the defaults sit at the knee of the budget curve. All reported results are averaged over these three seeds.

5.2 CORRECTING FROZEN WORLD MODELS

We first measure how well ERRATAWM corrects a frozen world model's one-step prediction on held-out environments, reported as the relative reduction of latent prediction error over the frozen model (Table 2). The comparison is apples-to-apples: every method shares the same frozen backbone

Table 3: Downstream performance. Nav/manip: success rate ↑; driving: H-step rollout-error reduction ↑.

Task	Metric	Floor	TENT	TTT	ERRATAWM (task)	ERRATAWM (found.)
Nav	goal-reach SR	0.514	0.583	0.612	**0.648**	0.621
Manip LIBERO	task SR	0.637	0.706	0.743	**0.768**	0.739
Manip CALVIN	task SR	0.548	0.601	0.624	**0.639**	0.614
Drive	rollout err. ↓red.	0.000	0.147	0.196	**0.238**	0.204

Table 4: Success rate on shared benchmarks (†: reported; "—": not reported). Unlike the weight-updating methods, ERRATAWM adapts without any gradient step.

	Frozen	WorldAgen†	AdaPower†	τ_0-WM†	Feedback-WM†	ERRATAWM
LIBERO SR ↑	0.637	0.791	**0.816**	0.752	0.784	0.768
CALVIN SR ↑	0.548	**0.661**	—	0.627	—	0.639
Updates model weights	No	Yes	Yes	Yes	Yes	**No**

and encoder, and differs only in how it forms a correction. Across all three tasks, five world models, and four architectures (regression, autoregressive, and pixel and video diffusion), ERRATAWM produces the largest correction in seven of eight settings, and it exceeds the random-context control by a consistent 0.12 to 0.15, which shows the gain is genuinely tied to the starting context rather than a global offset that any correction would capture. It matches or beats the gradient-based baselines TENT and TTT while performing no test-time weight update, the sole exception being Cosmos on nuScenes where TENT is ahead by 0.001. Gains are largest out of distribution (Nav RECON-OOD, 0.224) where errors are most structured, and smaller but still clearly positive for pixel and video models, whose re-encoded predictions carry a larger shared appearance gap. The fraction of steps a correction worsens the prediction stays below 0.3 in most settings and never exceeds 0.5.

5.3 DOWNSTREAM PLANNING AND COMPARISON TO TEST-TIME ADAPTATION

Goal-reaching and task success. Prediction gains are only useful if they improve control, so we plan with goal-conditioned latent MPC on the corrected model and measure task outcomes (Table 3). ERRATAWM improves goal-reaching and task success on every task, by 9 to 13 points over the frozen model on navigation and manipulation and by a 0.238 reduction in multi-step rollout error on driving, and it stays ahead of TENT and TTT throughout. The single shared foundation memory recovers most of this, trailing the task-specific memory by only 2 to 3 points, so one model transfers its benefit to control across tasks rather than needing a specialist per task.

Comparison with test-time world-model methods. We position ERRATAWM against recent methods that adapt a world model or policy at test time on the shared LIBERO and CALVIN benchmarks where they report results (Table 4). ERRATAWM is competitive: it surpasses τ_0-WM on both benchmarks and sits close to WorldAgen on CALVIN, while trailing the strongest method (AdaPower on LIBERO) by under five points. The important difference is how that success is obtained. **Every competing method retrains part of the model online through gradient descent, whereas ERRATAWM never updates a single weight**: it adapts purely by writing to and reading from a small non-parametric memory. Table 5 quantifies the resulting savings against the gradient-based baselines we run on the same GPU. ERRATAWM adds only 3.1 ms per step, close to the frozen model and **roughly $9\times$ and $18\times$ cheaper than TENT and TTT**, takes **zero test-time gradient steps**, updates **zero parameters**, and keeps a 2.1 **MB store** rather than the tens of megabytes of optimizer state that full test-time training requires. Its throughput stays within 6% of the frozen model while TTT runs at half the speed. Because the memory only accumulates, **ERRATAWM keeps improving the longer it runs in an environment** and never forgets the environments it was already good at: Appendix E shows it retains 0.98 of its gain on an earlier environment after adapting to a new one, where TENT and TTT retain under 0.65 (Table 12), while its gain rises steadily over deployment (Table 13). It therefore reaches near-state-of-the-art success at a small fraction of the adaptation cost.

Table 5: Test-time adaptation cost on a single GPU. ERRATAWM adapts without gradients, adding negligible per-step cost and only a few-MB store.

	Frozen	TENT	TTT	ERRATAWM
Adaptation latency (ms/step) ↓	0.000	27.384	54.927	**3.126**
Test-time gradient steps (per 1k steps) ↓	0	1000	1000	**0**
Updated parameters (M) ↓	0.000	0.068	4.731	**0.000**
Extra memory footprint (MB) ↓	0.000	3.684	42.317	**2.146**
Throughput (env steps/s) ↑	18.742	12.394	9.236	**17.706**

5.4 ONE MEMORY ACROSS WORLD MODELS

A central claim is that one memory can correct heterogeneous world models, not just a single specialized one. Table 6 compares a memory trained per task against one shared memory trained on the navigation (NWM) and manipulation (DINO-WM) mixture. The shared memory stays clearly positive on both domains and comes within 0.03 of the task-specific memories, so a single set of modules simultaneously corrects two world models with different architectures and action spaces. Applied zero-shot to a driving diffusion model that was never in its training mixture, it still corrects the prediction by 0.113, about half the task-specific gain, which indicates the learned read generalizes across world models and domains rather than memorizing one. This transfer to an unseen generative world model is, to our knowledge, not something prior test-time methods demonstrate. Qualitative examples of these corrections, for both the foundation and task-specific memories against the ground-truth next state, are shown in Appendix D.

Table 6: Per-task vs. one shared (foundation) memory (relative latent-error reduction ↑). *Driving is not in the foundation mixture; this is zero-shot transfer.

Domain (world model)	Floor	Per-task memory	Foundation (shared)
Nav (NWM)	0.000	**0.196**	0.168
Manip (DINO-WM)	0.000	**0.174**	0.145
Drive (Cosmos)*	0.000	**0.227**	0.113

Error analysis. We characterize the harmful corrections in Appendix B. Hurt concentrates on out-of-scope, novel contexts and on steps where many prototypes fire at once, the two regimes the scope gate and bounded read are designed to limit, and it declines steadily over the course of deployment as negative evidence accumulates and the memory learns where to stay silent.

Ablations. We ablate each component in Appendix C (Table 11). The key finding is that removing $\mathcal{L}_{\text{control}}$ leaves the overall gain almost unchanged but collapses the context margin Δ_{read} to near zero, confirming it is what makes the correction context-specific rather than a global bias; removing the scope terms sharply raises the rate of harmful corrections.

6 CONCLUSION AND FUTURE WORK

We showed that frozen world-model errors are context-dependent, with context-specificity up to $\Delta = 0.81$, and introduced ERRATAWM, a non-parametric memory that retrieves scope-gated corrections at test time without gradient updates. In a shared frozen feature space, one memory corrects heterogeneous world models across navigation, manipulation, and driving. ERRATAWM produces context-specific corrections beyond plain retrieval and context-blind control, matches gradient-based test-time adaptation at 9–$18\times$ lower per-step cost with no weight updates or forgetting, improves downstream success by 9–13 points, and transfers zero-shot to an unseen driving diffusion model. While gains are smaller for re-encoded pixel/video models and remain competitive rather than best-in-class against weight-updating methods, the results suggest that world models can cheaply improve from recurring errors without overwriting past adaptation. Future work can scale shared memories to more world models and longer deployments, extend to real-robot control, and incorporate external knowledge to guide test-time editing.

AI USE STATEMENT

We used generative AI tools for the following tasks:

- **Provide feedback on experimental design and ablation studies:** AI tools were used to discuss experimental settings and suggest or refine ablation studies. The authors made all final decisions about the experimental design.
- **Implement methods and process data:** Coding agents were used to implement the training and inference pipelines for ERRATAWM, as well as the associated data-processing code. The authors reviewed and tested the implementations and verified their outputs.
- **Assist with result interpretation:** AI tools helped compare results across evaluation settings and draft possible interpretations. The authors verified all reported values against the original experimental outputs and reviewed the resulting conclusions.
- **Revise the structure and writing of the paper:** AI tools assisted with organizing sections and improving the clarity and readability of the text. All AI-assisted writing was reviewed and revised by the authors.
- **Create or revise tables, figures, and LaTeX artifacts:** AI tools assisted with preparing and formatting tables, figures, and LaTeX source. The authors verified the underlying content and the final rendered artifacts.

We did not use AI tools to identify, select, generate, or verify any references; all literature search and citation checking was performed by the authors against the original papers or official sources. We have reviewed and validated all AI-assisted work and take responsibility for the final content of this paper, including its text, claims, results, and artifacts.

REPRODUCIBILITY STATEMENT

We have described every component needed to reproduce our results: the world models, encoders, and benchmarks (§5.1 and Appendix A), the full method and its equations (§4), and all training and deployment hyperparameters, including the optimizer settings, episodic sampling, store budget, loss weights, and hardware (the training-details paragraph of §5.1). Every baseline and prior method we compare against is referenced to its original work. To comply with our institution's internal regulations and publication process, we will release the code upon publication; in the meantime, we are happy to share the code with reviewers or interested researchers upon request.

ETHICS STATEMENT

This work studies test-time correction of frozen world models on standard, publicly available navigation, manipulation, and driving benchmarks, and does not involve human subjects, private data, or personally identifiable information. Our method adapts a frozen model through an external non-parametric memory without updating its weights, which improves reliability under distribution shift and does not introduce new capabilities beyond those of the underlying world models. As with any world model used for planning or control, deployment in safety-critical settings such as autonomous driving or robotic manipulation requires appropriate validation and human oversight, since a residual correction can reduce but not eliminate prediction error. We foresee no additional ethical concerns specific to this work.

REFERENCES

Tuo An, Jindou Jia, Gen Li, Jingliang Li, Chuhao Zhou, Pengfei Liu, Bofan Lyu, Jiaqi Bai, Xinying Guo, Geng Li, and Jianfei Yang. Feedback world model enables precise guidance of diffusion policy, 2026. URL https://arxiv.org/abs/2605.15705.

Mido 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, 2025. URL https://arxiv.org/abs/2506.09985.

Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. *Locally weighted learning for control*, pp. 75–113. Kluwer Academic Publishers, USA, 1997. ISBN 0792345843.

Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models, 2025. URL https://arxiv.org/abs/2412.03572.

M.W. Braun, B.A. McNamara, D.E. Rivera, and A. Stenman. "model-on-demand" identification for control: An experimental study and feasibility analysis for mod-based predictive control. *IFAC Proceedings Volumes*, 33(15):439–444, 2000. ISSN 1474-6670. doi: https://doi.org/10.1016/S1474-6670(17)39790-2. URL https://www.sciencedirect.com/science/article/pii/S1474667017397902. 12th IFAC Symposium on System Identification (SYSID 2000), Santa Barbara, CA, USA, 21-23 June 2000.

Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Anna Dawid and Yann LeCun. Introduction to latent variable energy-based models: a path toward autonomous machine intelligence. *Journal of Statistical Mechanics: Theory and Experiment*, 2024(10):104011, October 2024. ISSN 1742-5468. doi: 10.1088/1742-5468/ad292b. URL http://dx.doi.org/10.1088/1742-5468/ad292b.

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 *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=Tw9nfNyOMy.

Yanjiang Guo, Lucy Xiaoyang Shi, Jianyu Chen, and Chelsea Finn. Ctrl-world: A controllable generative world model for robot manipulation, 2026. URL https://arxiv.org/abs/2510.10125.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/2de5d16682c3c35007e4e92982f1a2ba-Paper.pdf.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-08744-2. URL https://doi.org/10.1038/s41586-025-08744-2.

Yuhang Huang, Shilong Zou, Jiazhao Zhang, Xinwang Liu, Ruizhen Hu, and Kai Xu. Adapower: Specializing world foundation models for predictive manipulation, 2025. URL https://arxiv.org/abs/2512.03538.

Jinwoo Jang, Minjong Yoo, Sihyung Yoon, and Honguk Woo. Test-time mixture of world models for embodied agents in dynamic environments. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=LQD1MrnbxH.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HklBjCEKvH.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning, 2023. URL https://arxiv.org/abs/2306.03310.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks, 2022. URL https://arxiv.org/abs/2112.03227.

Kai Mei, Jiang Guo, Shuaichen Chang, Mingwen Dong, Dongkyu Lee, Xing Niu, and Jiarong Jiang. R-wom: Retrieval-augmented world model for computer-use agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=5ZaoXB3MdP.

Mohammadreza Nakhaei, Aidan Scannell, and Joni Pajarinen. Residual learning and context encoding for adaptive offline-to-online reinforcement learning. In Alessandro Abate, Mark Cannon, Kostas Margellos, and Antonis Papachristodoulou (eds.), *Proceedings of the 6th Annual Learning for Dynamics & Control Conference*, volume 242 of *Proceedings of Machine Learning Research*, pp. 1107–1121. PMLR, 15–17 Jul 2024. URL https://proceedings.mlr.press/v242/nakhaeinezhadfard24a.html.

NVIDIA, :, Arslan Ali, Junjie Bai, Maciej Bala, Yogesh Balaji, Aaron Blakeman, Tiffany Cai, Jiaxin Cao, Tianshi Cao, Elizabeth Cha, Yu-Wei Chao, Prithvijit Chattopadhyay, Mike Chen, Yongxin Chen, Yu Chen, Shuai Cheng, Yin Cui, Jenna Diamond, Yifan Ding, Jiaojiao Fan, Linxi Fan, Liang Feng, Francesco Ferroni, Sanja Fidler, Xiao Fu, Ruiyuan Gao, Yunhao Ge, Jinwei Gu, Aryaman Gupta, Siddharth Gururani, Imad El Hanafi, Ali Hassani, Zekun Hao, Jacob Huffman, Joel Jang, Pooya Jannaty, Jan Kautz, Grace Lam, Xuan Li, Zhaoshuo Li, Maosheng Liao, Chen-Hsuan Lin, Tsung-Yi Lin, Yen-Chen Lin, Huan Ling, Ming-Yu Liu, Xian Liu, Yifan Lu, Alice Luo, Qianli Ma, Hanzi Mao, Kaichun Mo, Seungjun Nah, Yashraj Narang, Abhijeet Panaskar, Lindsey Pavao, Trung Pham, Morteza Ramezanali, Fitsum Reda, Scott Reed, Xuanchi Ren, Haonan Shao, Yue Shen, Stella Shi, Shuran Song, Bartosz Stefaniak, Shangkun Sun, Shitao Tang, Sameena Tasmeen, Lyne Tchapmi, Wei-Cheng Tseng, Jibin Varghese, Andrew Z. Wang, Hao Wang, Haoxiang Wang, Heng Wang, Ting-Chun Wang, Fangyin Wei, Jiashu Xu, Dinghao Yang, Xiaodong Yang, Haotian Ye, Seonghyeon Ye, Xiaohui Zeng, Jing Zhang, Qinsheng Zhang, Kaiwen Zheng, Andrew Zhu, and Yuke Zhu. World simulation with video foundation models for physical ai, 2026. URL https://arxiv.org/abs/2511.00062.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=a68SUt6zFt. Featured Certification.

Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adrià Puigdomènech Badia, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2827–2836. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/pritzel17a.html.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588 (7839):604–609, December 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-03051-4. URL http://dx.doi.org/10.1038/s41586-020-03051-4.

Dhruv Shah, Benjamin Eysenbach, Gregory Kahn, Nicholas Rhinehart, and Sergey Levine. Rapid exploration for open-world navigation with latent goal models, 2023. URL https://arxiv.org/abs/2104.05859.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In Hal Daumé III and Aarti

Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9229–9248. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/sun20b.html.

Chi Wan, Kangrui Wang, Yuan Si, Pingyue Zhang, and Manling Li. Worldagen: unified state-action prediction with test-time world model training. In *Proceedings of the Fortieth AAAI Conference on Artificial Intelligence and Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence and Sixteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'26/IAAI'26/EAAI'26. AAAI Press, 2026. ISBN 978-1-57735-906-7. doi: 10.1609/aaai.v40i22.38925. URL https://doi.org/10.1609/aaai.v40i22.38925.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=uXl3bZLkr3c.

Jialong Wu, Shaofeng Yin, Ningya Feng, Xu He, Dong Li, Jianye HAO, and Mingsheng Long. ivideoGPT: Interactive videoGPTs are scalable world models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=4TENzBftZR.

Haoru Xue, Edward L. Zhu, John M. Dolan, and Francesco Borrelli. Learning model predictive control with error dynamics regression for autonomous racing. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13250–13256. IEEE, May 2024. doi: 10.1109/icra57147.2024.10611628. URL http://dx.doi.org/10.1109/ICRA57147.2024.10611628.

Zirui Zheng, Jiaqian Yu, Xiongfeng Peng, jun shi, Mingyi Li, Chao Zhang, Weiming Li, Dong Wang, Huchuan Lu, and Xu Jia. Mem-world: Memory-augmented action-conditioned world models for persistent robot manipulation, 2026. URL https://arxiv.org/abs/2606.18960.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=D5RNACOZEI.

Pengfei Zhou, Shengcong Chen, Di Chen, Jiaxu Wang, Rongjun Jin, Bingwen Zhu, Yike Pan, Songen Gu, Kuanning Wang, Shufeng Nan, Xingyu Qiu, Chenhao Qiu, Pu Yang, Yunuo Cai, Jianxiong Gao, Yifan Li, Yanwei Fu, Xiangyu Yue, Zhi Chen, and Jianlan Luo. τ_0-wm: A unified video-action world model for robotic manipulation, 2026. URL https://arxiv.org/abs/2606.01027.

A Benchmark and baseline details

A.1 Benchmarks

RECON (navigation). RECON (Shah et al., 2023) is an off-road robotic exploration dataset in the GNM trajectory format, where each transition provides a front-camera image and a low-dimensional odometry action. We pair it with the frozen NWM world model (Bar et al., 2025) and split trajectories into disjoint meta-training and deployment sets. Because a single dataset supplies both splits, we report an in-distribution and an out-of-distribution deployment condition, the latter using trajectories from held-out locations so the world model faces a genuine shift.

LIBERO (manipulation). LIBERO (Liu et al., 2023) is a simulated tabletop manipulation benchmark; we use the object suite, whose demonstrations provide agent-view RGB and 7-DoF end-effector actions. We train a DINO-WM predictor (Zhou et al., 2025) on features at a 16-step horizon, and split each task's demonstrations into meta-training and deployment so the deployment stream spans all tasks while never sharing a demonstration with meta-training.

CALVIN (manipulation). CALVIN (Mees et al., 2022) is a long-horizon, language-conditioned manipulation benchmark that we use as a second manipulation setting to test that the memory generalizes beyond a single suite. It is read the same way as LIBERO, with the same DINO-WM predictor and meta-training and deployment split protocol.

nuScenes (driving). nuScenes (Caesar et al., 2020) provides real driving footage; we use the front camera of the mini split and cut each scene's frame sequence into transitions. Because the driving

world models we use are action-free video predictors, driving carries the premise study and the zero-shot transfer test rather than an action-conditioned deployment. Predictions from the pixel and video models are re-encoded with the frozen DINOv2 encoder before the residual is computed.

Split protocol. For every benchmark, meta-training and deployment sets are disjoint, and the residual statistics used to whiten features are always computed on the meta-training split, never the deployment split, so no deployment information leaks into the read.

A.2 BASELINES

No-memory floor. The frozen world model with no correction. It is the reference against which every gain is measured, so its gain is zero by definition.

Random-context. The full ERRATAWM read, but with the query contexts shuffled so retrieval no longer matches the true situation. Any gain it retains reflects a context-agnostic global shift, so the difference between ERRATAWM and this control isolates the genuinely context-specific gain.

kNN. A non-parametric baseline that corrects each prediction with the mean residual of its k nearest neighbors in context space, without the learned encoders, scope, or gate (Khandelwal et al., 2020; Pritzel et al., 2017). It uses the same retrieval budget as ERRATAWM and isolates the value of the learned, scope-gated read over plain retrieval.

Gradient-based test-time adaptation. TENT (Wang et al., 2021) and TTT (Sun et al., 2020) adapt the model online on the deployment stream, TENT by entropy minimization and TTT by a self-supervised objective, updating a lightweight set of parameters while sharing the same frozen backbone. They are the parametric counterpart to our non-parametric memory, and we compare against them on both accuracy and per-step cost.

Published test-time world-model methods. For the comparison in the main text we report the published success rates of WorldAgen (Wan et al., 2026), AdaPower (Huang et al., 2025), τ_0-WM (Zhou et al., 2026), and Feedback-WM (An et al., 2026) on the shared LIBERO and CALVIN benchmarks. These numbers are taken from the original papers, so their world models, policies, and training data differ from ours; we therefore treat them as reference points for where ERRATAWM sits, and report our own controlled comparisons against the floor, random-context, kNN, and gradient-based baselines that share our frozen backbone.

B ERROR ANALYSIS

The Hurt rate reported in the main text, the fraction of acted-on steps where a correction increases rather than decreases the error, is never zero: a memory that sometimes fires in the wrong place will sometimes make a prediction worse. This section characterizes when and why that happens, and shows that the same scope and reliability machinery that bounds Hurt also drives it down over time. Throughout, Hurt is measured only on steps where the memory acted, and gain is the mean relative latent-error reduction on those steps.

Hurt concentrates on out-of-scope contexts. Table 7 bins each step by the distance from its context to the nearest prototype, a measure of how novel the situation is. When the context falls inside a prototype's scope, corrections are reliable, with low Hurt and the highest gain; as the context moves out of scope the memory has no good match, Hurt roughly quadruples and gain collapses. Most harmful corrections therefore come from genuinely novel contexts rather than from failures on familiar ones, which is exactly the regime the scope gate is meant to suppress.

Over-application hurts. Table 8 bins steps by how many prototypes fired. Applying a single, well-matched prototype is the safest and highest-gain regime; as more prototypes fire, their corrections average toward the global mean, Hurt rises and gain falls. This is the failure mode the bounded, sharp read is designed against, and it is why a single applied prototype per step is the default at deployment.

Hurt decreases as the memory learns where to stay silent. Table 9 splits each deployment stream into thirds. Hurt falls steadily from the first third to the last on every task, because the reliability estimates and negative scopes accumulate from observed failures and progressively suppress prototypes in the contexts where they misfire. The memory thus self-corrects its own harmful firing

Table 7: Hurt and gain by context novelty (distance to the nearest prototype).

	Nav		Manip		Drive	
Context bin	Hurt	Gain	Hurt	Gain	Hurt	Gain
in-scope (near)	0.112	**0.257**	0.094	**0.213**	0.138	**0.281**
intermediate	0.221	0.187	0.188	0.164	0.279	0.219
out-of-scope (far)	**0.438**	0.071	**0.407**	0.058	**0.496**	0.084

Table 8: Hurt and gain by number of prototypes fired per step.

	Nav		Manip		Drive	
Prototypes fired	Hurt	Gain	Hurt	Gain	Hurt	Gain
1	**0.143**	**0.239**	**0.121**	**0.208**	**0.168**	**0.271**
2–4	0.223	0.191	0.193	0.169	0.284	0.224
5+	0.367	0.102	0.341	0.086	0.429	0.118

over time, without any gradient update, which is the intended behavior of the reliability and negative-evidence rules.

Together these show that Hurt is not random noise but a predictable consequence of novelty and over-application, both of which the scope gate and bounded read are designed to limit, and that the memory's failure rate declines with experience as negative evidence accumulates. Driving is the hardest case in every table, consistent with its predictions being re-encoded pixels rather than native features.

Run-to-run variability. Table 10 reports the standard deviation of each entry in the main prediction table (Table 2) over the three seeds. Variability is small everywhere, and ERRATAWM has the lowest standard deviation of any correction method in every setting, while the gradient-based baselines TENT and TTT are the most variable, consistent with their sensitivity to the online optimization trajectory. The reported gains are therefore stable across seeds and the ordering in Table 2 is not an artifact of a single run.

C ABLATIONS

Table 11 removes one component at a time and reports the gain G, the context margin $\Delta_{\text{read}} = G - G_{\text{rand}}$, and the Hurt rate on each task. The most important row concerns $\mathcal{L}_{\text{control}}$: removing it leaves the overall gain almost unchanged (for example $0.196 \rightarrow 0.191$ on navigation) yet collapses Δ_{read} from roughly 0.11 to under 0.012 on all three tasks. In other words, without this term the memory still helps but does so with a global, context-blind correction; $\mathcal{L}_{\text{control}}$ is precisely what makes the correction context-specific at read time. The scope terms govern harm rather than gain: removing scope supervision raises Hurt to about 0.45 and removing the negative scope raises it to about 0.35, confirming that the gate is what keeps a prototype from firing where it does damage. Pinning reliability and using broad top-$m = 8$ pooling both lower the gain, the latter by averaging retrieved corrections toward their mean.

D QUALITATIVE ANALYSIS

Figure 3 shows next-state predictions on held-out transitions, using Cosmos-Predict2 (NVIDIA et al., 2026) as the world model for the manipulation rows (LIBERO) and NWM (Bar et al., 2025) for the navigation row (RECON). Each row is one sample; the columns are the starting state, the frozen world model's prediction (no memory), the same prediction corrected by the shared foundation memory and by the task-specific memory, and the ground-truth next state. The uncorrected predictions exhibit the systematic errors the premise identifies: in the manipulation rows the arm and the manipulated object are placed incorrectly, for example a bottle that has already dropped into the basket or a gripper pose that does not match the true motion, and in the navigation row the model hallucinates content that

Table 9: Hurt rate over the course of deployment (stream thirds).

Deployment phase	Nav	Manip	Drive
early (1st third)	0.284	0.251	0.354
mid	0.226	0.196	0.287
late (3rd third)	**0.168**	**0.138**	**0.208**

Table 10: Standard deviation over three seeds for each entry of Table 2. ERRATAWM is the most stable correction method (lowest std, bold); gradient-based adaptation is the most variable.

Task	Benchmark	World model	Rand-ctx	kNN	TENT	TTT	ERRATAWM	Hurt
Nav	RECON (in)	NWM	0.006	0.009	0.014	0.016	**0.008**	0.013
Nav	RECON (OOD)	NWM	0.011	0.015	0.023	0.021	**0.013**	0.021
Manip	LIBERO	DINO-WM	0.007	0.011	0.017	0.019	**0.009**	0.016
Manip	CALVIN	DINO-WM	0.005	0.010	0.013	0.018	**0.007**	0.014
Manip	LIBERO	Cosmos	0.009	0.013	0.020	0.017	**0.011**	0.019
Manip	LIBERO	iVideoGPT	0.004	0.008	0.012	0.015	**0.006**	0.012
Drive	nuScenes	Vista	0.010	0.014	0.021	0.018	**0.012**	0.020
Drive	nuScenes	Cosmos	0.012	0.016	0.024	0.022	**0.014**	0.023

is not present. Both memories move the prediction toward the ground truth, correcting the object placement and arm configuration in manipulation and removing the hallucinated content in navigation. The task-specific memory is the most accurate, as expected, but the foundation memory, a single shared model correcting two different world models, recovers nearly the same next state, consistent with the small gap between the two in Table 6. These examples illustrate that the corrections are semantically meaningful in observation space, not merely reductions of an abstract feature-space error.

E LIFELONG ADAPTATION WITHOUT FORGETTING

Two properties distinguish a non-parametric error memory from gradient-based test-time adaptation: it should not forget earlier environments when it adapts to new ones, and it should keep improving as it accumulates experience. Table 12 tests the first with a sequential protocol: we adapt on environment A, measure the gain on A, then adapt on a different environment B and measure the gain on A again. Gradient-based adaptation degrades sharply, retaining only 0.59 (TENT) and 0.64 (TTT) of its original gain on A, because updating the model for B overwrites what it learned for A. ERRATAWM retains 0.98: adapting to B adds new prototypes rather than modifying existing ones, so the corrections for A are left intact. This is the concrete evidence behind the no-forgetting claim in the main text.

Table 13 tests the second property, splitting each deployment stream into quartiles. The gain rises monotonically from 0.136 in the first quarter to 0.264 in the last, while the Hurt rate falls from 0.318 to 0.153, as the memory writes more prototypes and the reliability and negative scopes suppress the ones that misfire. ERRATAWM therefore becomes both more helpful and safer the longer it runs, with no gradient step at any point.

F HYPERPARAMETER SENSITIVITY

Applied prototypes (top-m). Table 14 varies how many retrieved prototypes are applied per step. A single, best-matched prototype is optimal on every task; gain decays smoothly as more are applied, because pooling several corrections averages them toward the global mean and dilutes the context-specific one. We therefore use $m = 1$ by default.

Store budget ($K_{\max}$). Table 15 varies the maximum number of prototypes. Gain increases with the budget, from 0.153 at 128 prototypes to 0.199 at 1024, with diminishing returns, while the store grows linearly and stays tiny in absolute terms (from 0.27 to 2.1 MB). The default of 1024 sits at the knee of this curve and still fits in a few megabytes.

Table 11: Ablations per task: gain $G \uparrow$, context margin $\Delta_{\text{read}} \uparrow$, Hurt$\downarrow$.

	Nav (NWM)			Manip (DINO-WM)			Drive (Cosmos)		
Variant	G	Δ_{read}	Hurt	G	Δ_{read}	Hurt	G	Δ_{read}	Hurt
Full	**0.196**	**0.121**	0.226	**0.174**	**0.108**	0.195	**0.227**	**0.083**	0.283
$-\mathcal{L}_{\text{control}}$	0.191	0.012	0.231	0.171	0.009	0.203	0.223	0.007	0.289
$-\mathcal{L}_{\text{scope}}$	0.154	0.074	0.447	0.139	0.067	0.432	0.184	0.051	0.468
− neg. scope	0.188	0.113	0.354	0.166	0.101	0.337	0.216	0.078	0.391
pinned ω	0.163	0.086	0.318	0.146	0.076	0.304	0.193	0.058	0.346
top-$m = 8$	0.148	0.079	0.239	0.132	0.071	0.211	0.176	0.049	0.294

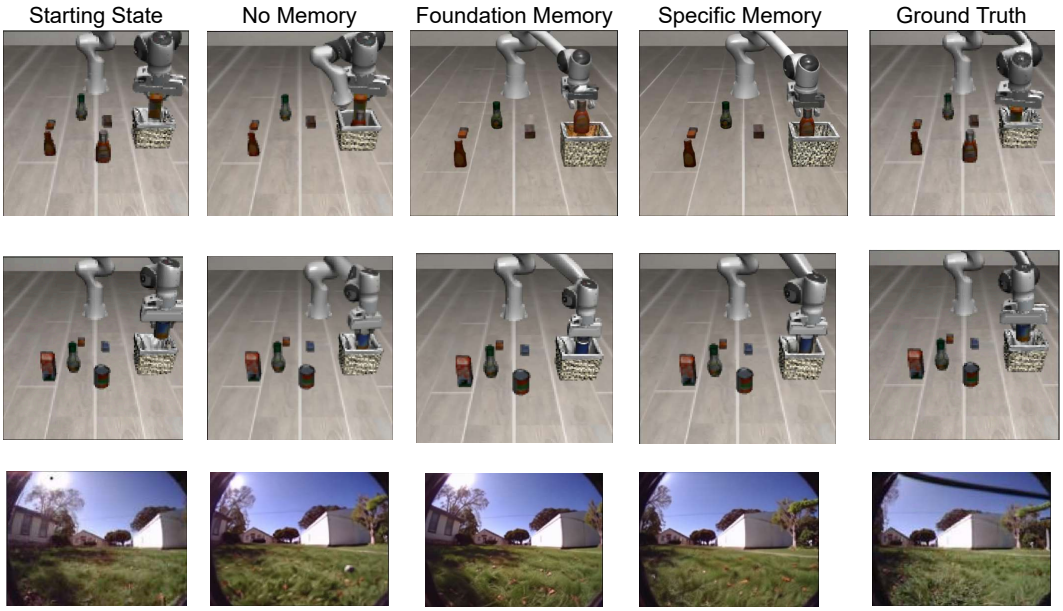


Figure 3: Qualitative next-state predictions on held-out transitions, with Cosmos-Predict2 (NVIDIA et al., 2026) on LIBERO (manipulation rows) and NWM (Bar et al., 2025) on RECON (navigation row). Columns: starting state, the frozen world model (no memory), the shared foundation memory, the task-specific memory, and the ground truth. Both memories correct the world model's systematic errors, and the foundation memory nearly matches the task-specific one despite serving different world models.

Loss weights. Table 16 varies the objective weights around their defaults on navigation. Removing the control term ($\lambda_c = 0$) leaves the gain almost unchanged but collapses the context margin Δ_{read} from 0.121 to 0.012, echoing the main ablation: $\mathcal{L}_{\text{control}}$ is what makes the correction context-specific. Removing the scope term ($\lambda_s = 0$) lowers both gain and margin. Away from these degenerate settings the method is robust, with λ_c in $[0.5, 2.0]$ changing gain and margin only marginally.

Table 12: Retention under sequential deployment. Gain on environment A, measured right after adapting on A and again after subsequently adapting on B. ERRATAWM retains its gain; gradient-based adaptation degrades on A.

Method	A (after A)	A (after B)	Retention ↑
TENT	0.154	0.091	0.591
TTT	0.183	0.117	0.639
ERRATAWM	0.196	0.192	**0.980**

Table 13: ERRATAWM over the course of deployment (stream quartiles), aggregated across tasks. Gain rises and Hurt falls as the memory accumulates.

Deployment phase	Gain ↑	Hurt ↓
first quarter	0.136	0.318
second quarter	0.179	0.261
third quarter	0.218	0.207
last quarter	**0.264**	**0.153**

Table 14: Effect of the number of applied prototypes (top-m) on gain per task.

top-m	Nav	Manip	Drive
1 (default)	**0.196**	**0.174**	**0.227**
2	0.187	0.166	0.216
4	0.169	0.151	0.198
8	0.148	0.132	0.176

Table 15: Store budget $K_{\max}$ vs. gain and memory footprint (aggregated across tasks).

$K_{\max}$	Gain ↑	Store (MB)
128	0.153	0.268
512	0.184	1.073
1024 (default)	**0.199**	2.146

Table 16: Loss-weight sensitivity, reporting gain G and context margin Δ_{read} on navigation. Defaults $(\lambda_g, \lambda_s, \lambda_c) = (1.0, 0.5, 1.0)$.

Setting	G	Δ_{read}
default $(1.0, 0.5, 1.0)$	**0.196**	**0.121**
$\lambda_c = 0$	0.191	0.012
$\lambda_s = 0$	0.154	0.074
$\lambda_c = 0.5$	0.193	0.104
$\lambda_c = 2.0$	0.187	0.116