Space-Aware World Models: Spatial Persistence Through Factored Scene Representations

Bruno Corcuera **Carlos Eiras-Franco** **Brais Cancela**

CITIC Research Center, Universidade da Coruña

{bruno.sanchez1,carlos.eiras.franco,brais.cancela}@udc.es

Abstract

Long-horizon navigation simulations require spatial persistence; revisited physical states must reproduce the same landmarks, rather than merely locally plausible frames. Standard latent-dynamics world models typically evolve a single predictive state that carries the scene layout, appearance, and agent motion simultaneously, which makes revisitation consistency brittle under compounding errors. We study a structured regime for navigation-like tasks in which the scene layout remains static during an episode, a map-like scene observation is available, and the agent motion can be described by a low-dimensional state. For this regime, we propose Space-Aware World Models (SWM). SWM factors world modeling into: (i) a scene latent inferred once per episode from a top-down map; (ii) an explicit, action-conditioned transition model over agent state; and (iii) a conditional renderer from scene code and state to observations. By anchoring scene content in an episode-level representation, SWM is designed to preserve the identity of revisited locations across autoregressive imagined trajectories. Across procedurally generated environments, SWM improves long-horizon fidelity and yields stronger revisitation and geometric consistency, as well as control-relevant fidelity. A matched entangled-latent control receives the same scene and state information but loses place identity over recurrent updates, including at a stationary pose. These results support scene-state factorization as an effective inductive bias for spatially coherent simulations.

1 Introduction

World models encode raw sensory information into compact representations, allowing agents to anticipate future observations and optimize policies without costly real-world interactions [1].

Most current world models learn an evolving latent state from past observations and actions, instantiated with recurrent state-space models [2, 3], transformers [4, 5], or diffusion-based predictors [6, 7]. This formulation is powerful but places the scene layout, appearance, and agent motion inside a shared predictive representation.

Embodied spatial prediction exposes a specific weakness in this formulation. Long-horizon simulation requires landmarks and layout to remain tied to the physical state: when an agent returns to a previously visited state, the generated observation should still correspond to the same place. We refer to this requirement as spatial persistence. The relevant failure mode is not simply poor short-horizon image quality, but loop-closure inconsistency: a rollout may contain locally plausible frames while rendering a different scene when queried at nearly the same state.

Scaling improves short-horizon realism but does not, by itself, impose this constraint. Larger world models may still rely on an evolving predictive state to preserve both scene structure and agent evolution. This can appear as plausible generated views that no longer match the scene implied by the

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

agent's state. This motivates a complementary factorization that separates persistent scene memory from state evolution [8, 9].

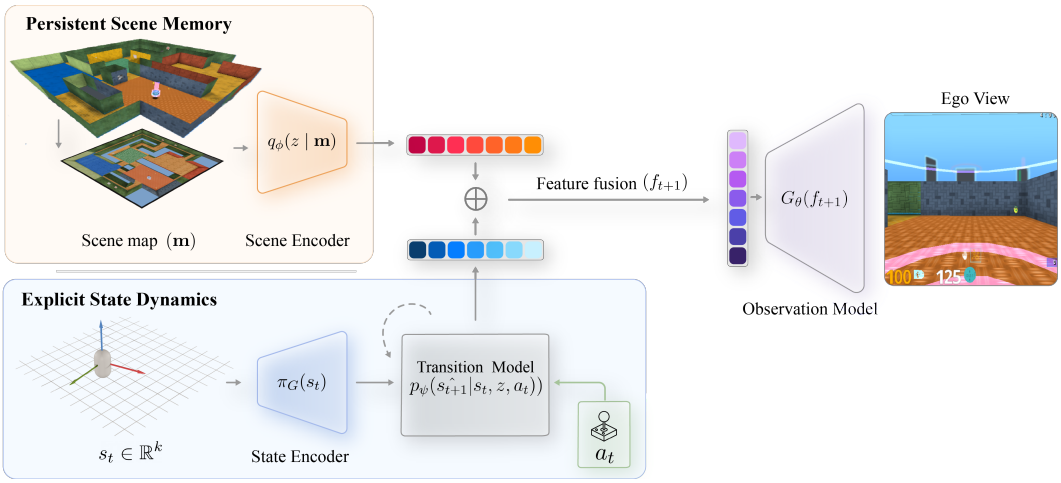


Figure 1: **Space-Aware World Model:** A scene latent $z \in \mathbb{R}^d$ is inferred once per episode from a top-down environment map $\mathbf{m}$, or sampled from the prior to generate synthetic scene instances. At each time step, the agent state $s_t \in \mathbb{R}^k$ is mapped through a Fourier feature transform to a state representation $\pi_s(s_t) \in \mathbb{R}^d$. The observation model fuses scene and state through a learned convex combination, $f_t = \alpha\, z + (1-\alpha)\, \pi_s(s_t)$, and a deterministic decoder maps f_t to the egocentric observation $\hat{o}_t$. The transition module predicts action-conditioned state deltas to evolve s_t over time. During rollout, the fixed scene code z anchors episode-level scene content while temporal prediction updates only the explicit state, providing a structural bias for spatial persistence.

We propose the Space-Aware World Model (SWM), a factored world model for map-conditioned embodied domains with a static intra-episode layout and low-dimensional agent state. SWM represents each scene with a persistent scene latent z, inferred from a top-down map or sampled from the learned prior, and an explicit agent state s_t that evolves under an action-conditioned transition model. A conditional renderer maps (z, s_t) to egocentric observations. By storing scene content in a persistent code and restricting temporal prediction to explicit state evolution, SWM turns long-horizon coherence from an implicit memory burden into a representational design choice [10, 11].

We assume that map-like scene observation and low-dimensional state supervision are available, and that the scene layout is approximately static within an episode. We use top-down maps as a controlled source of scene information, but the scene–state factorization is not inherently map-dependent. We evaluate this setting in a compact model regime, where all systems are trainable on a single workstation. A matched entangled-latent control receives the same scene and state information, allowing us to test whether retaining a fixed, separately accessible scene code improves persistence beyond the information advantage alone.

Under this factorization, world modeling becomes closer to viewpoint-conditioned rendering driven by explicit state evolution than to unconstrained video prediction [12, 13]. The same structure enables a sharper evaluation: because the physical state is explicit, we can test whether the generated observations remain consistent under revisitation, loop closure, known-pose geometry, and downstream control probes [14, 15].

We evaluate SWM in two domains: a procedurally generated 3D navigation environment that stresses revisitation, loop closure, and geometry-aware consistency, and a continuous-control racing environment that probes whether dreamed observations preserve control-relevant information for a fixed downstream policy. We also compare SWM with a matched entangled-latent control that receives the same map code and initial state, isolating the factorization from the additional information. The implementation is available on GitHub[1].

[1] `https://github.com/BrunooCS/space-aware-world-models`

Contributions.

1. We formulate a map-conditioned world model for navigation-like domains with static intra-episode layout and low-dimensional state, explicitly separating persistent scene memory from action-conditioned state evolution.
2. We introduce diagnostics for spatial persistence, including matched-state revisitation, return-to-origin panorama consistency, known-pose epipolar consistency, and agreement with a frozen downstream policy.
3. We show improved long-horizon spatial coherence over an entangled-latent control with matched information and comparable recurrent capacity, with supporting spatial and control-relevant diagnostics against compact latent-dynamics baselines trained on the same data and workstation-scale compute budget.

2 Related Work

Latent-Dynamics World Models. Early world models, such as *World Models* [1, 2] and PlaNet [16], established the template of latent-dynamics for imagination-based control, while the Dreamer family refined recurrent state-space modeling for model-based reinforcement learning [3, 17–19]. Recent approaches strengthen this template with transformers, diffusion models, and joint-embedding predictors, as in IRIS, DIAMOND, and LeWM [4, 6, 20]. At a larger scale, models such as GAIA-1, Navigation World Models, and V-JEPA 2 show that high-capacity architectures and large data can produce powerful predictive representations [21, 7, 22]. In these models, long-horizon prediction is typically mediated by an evolving learned state updated through rollout. SWM studies a complementary structured regime: when a static layout observation and low-dimensional pose are available, scene content can be encoded as persistent episode context while dynamics update explicit state. This separates SWM from generic video-only world modeling and frames the comparison as an inductive-bias study rather than an equal-input benchmark.

Neural Scene Representations and Conditional Rendering. The closest conceptual precedent for the SWM observation model is viewpoint-conditioned scene rendering. The Generative Query Network (GQN) showed that a compact scene representation and a query viewpoint can be provided separately to a learned renderer, enabling novel-view generation conditioned on a scene summary [12]. Consistent GQN [23] further showed that coherence across queried views benefits from sharing scene-level latent information. SWM builds on this renderer-level separation but uses it for a different purpose: the scene code is inferred once from a top-down map, held fixed as episode context, and combined with action-conditioned state evolution to support spatially grounded rollouts.

Neural rendering methods such as NeRF condition view synthesis on camera pose and scene structure, and Fourier positional encodings are a standard mechanism for enabling high-frequency spatial conditioning in such models [24, 25]. SWM is aligned with this perspective: it treats observation prediction as conditional rendering from scene context and pose, and applies Fourier features to the agent's kinematic state before decoding. The main distinction is that classical NeRF-style methods are typically optimized for a particular scene, whereas SWM learns a scene-code distribution across environments and uses each sampled or inferred code as persistent context for an action-conditioned rollout. NeRF-VAE and SRT infer scene representations from input views [26, 27], while LVSM performs novel-view synthesis from sparse posed views [28]. These approaches complement SWM by addressing how scene information is obtained, whereas our focus is on preserving it through action-conditioned rollouts.

Spatial Memory and Neural Mapping. A related line of work equips embodied agents with explicit or external memory for navigation, including spatially organized neural maps, learned SLAM-style memory, and attention over stored scene observations [29–32]. These methods address long-horizon partial observability primarily at the policy, mapping, or memory level. SWM is complementary: it places persistent scene information inside the generative simulation, pairs it with explicit action-conditioned state evolution, and evaluates the resulting simulations through revisitation, loop-closure, and geometry-aware consistency. Recent video and driving world models also incorporate memory, retrieval, or geometry-grounded mechanisms to reduce long-horizon drift [33–36]. SWM focuses on the specific case where a static scene description is available and can be used as episode-level context throughout the simulation.

3 Space-Aware World Model

We propose a Space-Aware World Model (SWM) for navigation-like environments where a map-like scene observation is available and agent motion is naturally represented in a low-dimensional state space. SWM factorizes prediction into three components (Figure 1): a scene encoder that maps the top-down environment view $\mathbf{m}$ to a persistent latent code $z \in \mathbb{R}^d$ inferred once per episode, an action-conditioned transition model over explicit agent state, and a conditional observation model that renders egocentric observations from the scene code and state. The factorization separates episode-level scene information from temporal state evolution.

Formally, given a top-down map $\mathbf{m}$, state s_t, action a_t, a state encoder π_s, and an action encoder π_a, SWM defines

$$\text{Scene encoder:} \quad z \sim q_\phi(z \mid \mathbf{m}) = \mathcal{N}(\mu_\phi(\mathbf{m}), \Sigma_\phi(\mathbf{m}))\,, \tag{1}$$

$$\text{Observation model:} \quad \hat{o}_t = G_\theta(z, \pi_s(s_t)), \tag{2}$$

$$\text{Transition model:} \quad s_{t+1} = s_t + T_\psi(\pi_s(s_t), z, \pi_a(a_t)). \tag{3}$$

During rollout, SWM encodes $\mathbf{m}$ once, initializes from the current state, and alternates between observation rendering and state transition. Thus, scene memory remains fixed while autoregressive prediction acts on explicit state. The experiments test whether this structural bias improves spatial consistency under long imagined trajectories.

3.1 Scene Encoder

The scene encoder maps the top-down environment view $\mathbf{m}$ to a latent scene code $z \in \mathbb{R}^d$. Its role is to capture persistent scene layout and appearance in a representation that is inferred once per scene and reused throughout the simulation. The current implementation uses a variational autoencoder [37]; architectural details are in Appendix C.

The variational formulation regularizes the scene representation and allows held-out maps to be embedded into a shared latent space without per-scene optimization. It also makes the renderer compatible with scene codes inferred from maps or sampled from the learned prior, $z \sim \mathcal{N}(0, I)$, as illustrated qualitatively in Section 4.6. This differs from scene-specific neural rendering methods such as NeRF, which typically optimize scene representations from posed observations for each scene.

3.2 Observation Model

The observation model G_θ is a deterministic renderer conditioned on a persistent scene code and the current agent state. The state pathway applies a Fourier feature transform [25, 38] followed by a linear projection to obtain $\pi_s(s_t) \in \mathbb{R}^d$, while the scene code enters the fusion space directly. The two pathways are combined by bounded additive fusion,

$$f_t = \alpha\, z + (1 - \alpha)\, \pi_s(s_t) \qquad \alpha \in [\epsilon, 1 - \epsilon], \quad 0 \le \epsilon < \tfrac{1}{2}. \tag{4}$$

The mixing coefficient α is one learned global scalar per domain, shared across scenes and episodes and applied uniformly across the fusion vector. We use $\epsilon = 0.05$ in both domains without metric-specific tuning. The bound prevents either fusion weight from vanishing, discouraging state-only shortcuts. The fused representation is decoded as $\hat{o}_t = G_\theta(z, \pi_s(s_t)) := \mathcal{D}_\theta(f_t)$, where $\mathcal{D}_\theta$ denotes the post-fusion image decoder.

This design serves two purposes. Empirically, scene information is supplied to the renderer at every time step rather than being carried through recurrent dynamics. Analytically, additive fusion preserves an explicit decomposition between scene and state contributions before decoding, used by diagnostics in Section 4 and formalized in Appendix A. Alternative fusion schemes, such as concatenation, may achieve comparable reconstruction fidelity but do not preserve the same closed-form fusion-space identities.

The observation model is trained with a weighted sum of pixel reconstruction and perceptual loss [39]. Decoder architecture details are in Appendix C.

Algebraic consequences of additive fusion. The additive interface gives two simple identities used by the diagnostics. First, with a fixed scene code and deterministic renderer, revisiting the same explicit state yields the same renderer input and output: $s_i = s_j \Rightarrow G(z, \pi_s(s_i)) = G(z, \pi_s(s_j))$. This is an input-level invariance, not a guarantee that an autoregressive rollout will return to the exact physical state. Second, for $f(z, s) = \alpha z + (1 - \alpha)\pi_s(s)$, fusion-space error separates into a fixed scene-code term and a time-varying state term:

$$f(\hat{z}, \hat{s}_t) - f(z^\star, s_t^\star) = \alpha(\hat{z} - z^\star) + (1 - \alpha)(\pi_s(\hat{s}_t) - \pi_s(s_t^\star)).$$

This identity is pre-decoder and does not imply an additive pixel-space error decomposition. It motivates the oracle-state analysis: replacing $\hat{s}_t$ with $s_t^\star$ removes accumulated transition drift while leaving scene-encoding and rendering error. Appendix A gives the formal statements, including the closed-form fusion-space compensation used in Appendix B.7.

3.3 Transition Model

The transition model predicts action-conditioned state increments with the persistent scene code as context:

$\Delta s_t = T_\psi(\pi_s(s_t), z, \pi_a(a_t))$, yielding $s_{t+1} = s_t + \Delta s_t$. Conditioning on z allows the same action-state pair to have different effects depending on the episode-specific geometry encoded by the scene representation. The pair (z, s_t) provides information about local traversability near the current state, so similar actions can produce different increments in open regions, near walls, or along curved tracks.

We predict state deltas rather than absolute next states, a natural choice in the considered domains where consecutive motions are small relative to the spatial extent of the environment. Delta prediction also makes rollout error easier to interpret geometrically: autoregressive drift appears as incremental mislocalization in explicit state space rather than as displacement in an opaque latent trajectory.

3.4 Training

We train the scene encoder, observation model, and transition model sequentially. First, the scene encoder is fit on top-down maps. Next, the observation model is trained on tuples (z, s_t, o_t) with precomputed reparameterized posterior scene samples rather than posterior means alone. Finally, the transition model is trained on action-conditioned state sequences with z as scene context.

We adopt staged training for diagnostic clarity, not as a claim of universal optimality. It lets us evaluate scene encoding, rendering, and dynamics separately. Generalization to held-out environments is an empirical result: the encoder maps unseen maps into the learned scene-latent space, and the renderer and transition model are evaluated on those codes without per-scene optimization.

4 Experiments

We evaluate the paper's central hypothesis: in domains with persistent scene structure and low-dimensional agent kinematics, separating scene memory from explicit state evolution provides a useful inductive bias for long-horizon spatial coherence. A rollout can appear locally plausible while losing the identity of a revisited place, so the main evaluation emphasizes state-conditioned, panorama-conditioned, and geometry-aware consistency rather than adjacent-frame fidelity alone.

Both domains use procedurally generated episodes with held-out layouts. DeepMind Lab (DMLab) [40] is a 3D first-person navigation benchmark that stresses revisitation, return-to-origin consistency, and spatial memory. CarRacing [41] is a continuous-control probe that tests whether the same factorization preserves track affordances used by a fixed downstream controller. The primary factorization test is an equal-information comparison with a matched entangled-latent control in DMLab (Section 4.5). Cross-family comparisons provide complementary evidence but do not isolate the effect of factorization from the additional scene and state information available to SWM.

4.1 Experimental setup

We train on 2×10^4 procedurally generated episodes, totaling roughly 10^7 frames, with one top-down environment image per episode. We compare against a matched entangled-latent control and

four latent-dynamics model families: an MDN-RNN world model [2], a causal transformer world model inspired by IRIS [4], DreamerV3 as an RSSM-based model [18], and LeWM as a JEPA-style embedding predictor [20]. All systems use the same rollout data and action sequences. Capacity and checkpoints are selected without test-set tuning; training continues until held-out validation loss plateaus. Baseline interfaces are detailed in Appendix C.5.

The main DMLab results focus on revisitation, return-to-origin panorama consistency, distributional support, and epipolar geometric consistency. The main CarRacing results focus on state-conditioned track consistency and frozen-policy agreement. We report horizon-resolved diagnostics in the main text and revisitation-gap curves in Appendix B.1. Full metric definitions, panorama decompositions, and qualitative examples are reported in Appendix B; optimization schedules and hyperparameters are reported in Appendix C. Unless noted, $\pm$ denotes standard deviation across held-out episodes, not training seeds.

4.2 DMLab: first-person navigation

DMLab is the main testbed for spatial persistence. In first-person maze navigation, standard frame metrics can be misleading: a model may generate a plausible corridor at every step while failing loop closure, meaning that a revisited physical state no longer produces the corresponding view. We therefore treat frame-wise rollout fidelity as supporting evidence and emphasize state and panorama-conditioned consistency.

We evaluate 100 held-out imagined rollouts of length 500. The primary revisitation metric is *State-match LPIPS*. Using matched physical states from the evaluation trajectory, it measures whether generated observations at revisited states remain perceptually close to the corresponding real views. Low *State-match LPIPS* means that the model preserves a stable association between state and appearance across temporal gaps, rather than only producing smooth adjacent frames.

The stricter loop-closure test is a return-to-origin 360° panorama benchmark. For each episode, we render a 36-view yaw sweep, 36 ego-views from the same position with headings spaced every 10° around the full circle. We render one sweep at the start pose and another after a matched 250-step interval. The *out-and-back* regime moves away from the start and returns near it, so success requires both path integration and memory of the original place. In the *stationary-control* regime the model receives 250 exact no-op actions while the ground-truth position and heading remain fixed, followed by the same in-place yaw sweep; it asks whether the model preserves the surrounding scene even when path integration should be nearly trivial.

Table 1: Cross-family DMLab diagnostics on 100 held-out dreamed rollouts of length 500. The table emphasizes diagnostics that test persistent place identity under long autoregressive rollout. The panorama metrics use the out-and-back return-to-origin benchmark with 36-view panoramas and a matched 250-step separation between departure and return. Rel. Imp. reports the relative improvement of SWM over the best latent-dynamics baseline for each metric. Standard frame-wise rollout metrics are reported in Table 5 and Figure 3.

Metric	MDN-RNN	Transformer	DreamerV3	LeWM	SWM	Rel. Imp. (%)
State-match LPIPS ↓	0.208 ± 0.115	0.168 ± 0.140	0.269 ± 0.185	0.407 ± 0.142	$\mathbf{0.033 \pm 0.061}$	**+80%**
Origin-match pano LPIPS ↓	0.545 ± 0.098	0.523 ± 0.098	0.593 ± 0.070	0.613 ± 0.094	$\mathbf{0.298 \pm 0.117}$	**+43%**
Origin pano Recall@1 ↑	0.070 ± 0.255	0.100 ± 0.300	0.000 ± 0.000	0.010 ± 0.099	$\mathbf{0.770 \pm 0.421}$	**+670%**
Pano yaw error (deg) ↓	74.6 ± 61.5	79.3 ± 63.6	78.4 ± 68.4	89.5 ± 57.7	$\mathbf{31.4 \pm 25.4}$	**+57%**
PRDC Density ↑	0.0659	0.1462	0.0057	0.0253	**0.3710**	**+154%**
PRDC Coverage ↑	0.1024	0.2078	0.0086	0.0402	**0.5288**	**+154%**
ECS Inlier Ratio ↑	0.2176	0.2396	0.0521	0.2878	**0.3507**	**+22%**

We use three panorama metrics. *Origin-match pano LPIPS* compares each predicted return view to the corresponding real start view at the same yaw bin, measuring strict visual fidelity, including orientation. *Origin pano Recall@1* retrieves the predicted return panorama's episode from a gallery of real start panoramas, testing place identity rather than generic perceptual plausibility. *Pano yaw error* reports the angular offset of the best circular perceptual match. It diagnoses orientation drift when place identity is retained; yaw error alone is not evidence of persistence when retrieval fails. These metrics address distinct failures, while out-and-back and stationary are control regimes. Definitions and auxiliary diagnostics are in Appendix B.4.

We also report PRDC density and coverage [42] and ECS Inlier Ratio. PRDC is computed in perceptual feature space: density measures whether generated samples fall in high-density regions of the real data manifold, while coverage measures how much of the real manifold is represented by generated samples. These metrics help distinguish faithful diverse generation from collapse to a small set of easy frames. ECS Inlier Ratio uses SIFT correspondences [43] and Sampson-error epipolar agreement [44] to test whether generated view pairs remain compatible with the known relative camera motion. Formal definitions are given in Appendix B.1.

Table 1 shows that SWM is strongest on diagnostics aligned with persistent place identity. It lowers State-match LPIPS and Origin-match pano LPIPS, and its Recall@1 indicates that the predicted return panorama is often identifiable as the correct held-out episode. The reduction in yaw error suggests that some remaining mismatch is attributable to orientation drift rather than complete replacement of the surrounding scene.

The supporting rows address two alternative explanations. First, PRDC density and coverage improve together, arguing against a collapse to a narrow set of easy-to-render frames. Second, ECS Inlier Ratio improves, indicating that the advantage is not purely perceptual: generated frame pairs more often support feature correspondences consistent with the known relative camera motion. These diagnostics characterize SWM's spatial fidelity; attribution to factorization relies on the matched control in Table 4. Near-zero DreamerV3 retrieval does not imply training collapse: locally plausible frames can fail to identify their original episode after a long prior-only rollout.

Recall@1 is binary per episode. At $p \approx 0.77$, an episode-level SD of $\sqrt{p(1-p)} \approx 0.420$ is expected; over 100 independent episodes, the corresponding standard error is approximately 0.042.

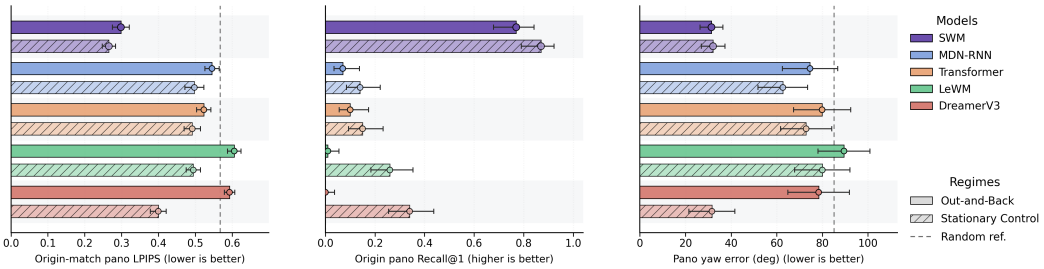


Figure 2: Return-to-origin panorama consistency on held-out DMLab environments. The out-and-back regime tests whether place identity is preserved after a long excursion and return. The stationary-control regime uses 250 exact no-op actions while keeping the same panorama rendering protocol. A large gap between the two regimes indicates accumulated transition drift, whereas weak performance in both regimes indicates that the model fails to preserve a stable scene representation even before substantial path integration is required.

Figure 2 localizes the main failure modes. SWM remains stable in the stationary-control regime and degrades only moderately in the out-and-back regime, consistent with residual state-transition drift. The more important contrast is that several latent-dynamics baselines remain weak even under stationary control, where the ground-truth pose is fixed. This suggests that their errors are not solely path-integration failures: an entangled predictive state can lose or overwrite scene information over time even when the physical state changes little. Section 4.4 further tests this interpretation with SWM-oracle, which retains the same scene code and renderer but uses ground-truth evaluation states before rendering.

4.3 CarRacing: continuous control

CarRacing tests the same factorization in a different regime. The goal is not to reproduce the DMLab loop-closure benchmark, but to ask whether generated observations preserve local track affordances that matter for steering, throttle, and braking. Standard rollout metrics are reported in Appendix B.1; the main text focuses on two policy-relevant diagnostics and one state-conditioned consistency diagnostic.

We use a frozen pixel-based CarRacing controller g as an evaluator. The controller is not trained or adapted on generated frames. A plausible road with a misplaced curve can induce incorrect

steering or braking, so the primary probe is *Matched-frame policy agreement*: the same controller receives paired real and generated observations at corresponding timesteps. We report Action error ($D_g = \mathbb{E}_t \|g(\hat{o}_t) - g(o_t)\|_1$), steering-sign agreement, and Pearson steering correlation [45]. *Dream action statistics* compare aggregate behavior in dreamed and real rollouts. Action W_1 averages Wasserstein distances [46] over steer, gas, and brake marginals; Action JS averages Jensen–Shannon divergences [47] over their histograms; and Coupling error compares pairwise steering, throttle, and braking correlations. The first block measures paired decision agreement at matched timesteps, while the second measures aggregate behavior induced by dreams.

Table 2: Frozen-policy control probe in CarRacing. Dream action statistics compare the action distribution induced by a pretrained pixel controller in dreamed rollouts against real rollouts. Matched-frame policy agreement evaluates the same controller on real and generated observations at corresponding timesteps. All metrics use 5×10^4 controller evaluations.

	Matched-frame policy agreement			Dream action statistics		
Model	Action err. ↓	Steer sign ↑	Steer corr. ↑	Action W_1 ↓	Action JS ↓	Coupling err. ↓
SWM	**0.655 ± 0.469**	**0.958**	**0.891**	**0.023 ± 0.034**	**0.002 ± 0.001**	**0.007**
LeWM	1.367 ± 0.908	0.640	0.127	0.077 ± 0.096	0.019 ± 0.020	0.209
Transformer	1.423 ± 0.704	0.755	0.091	0.085 ± 0.077	0.018 ± 0.014	0.093
DreamerV3	1.446 ± 0.998	0.627	0.147	0.063 ± 0.054	0.014 ± 0.012	0.097
MDN-RNN	1.467 ± 0.789	0.624	-0.025	0.104 ± 0.098	0.023 ± 0.018	0.205

Table 2 shows that SWM outperforms other methods under both frozen-policy views. It closely matches the real-rollout action distribution and reduces action error while preserving steering direction at 95.8% timesteps. Since the controller is fixed, these metrics diagnose policy-facing observation fidelity rather than end-to-end policy learning.

Table 3: Supporting CarRacing spatial and distributional diagnostics. State-match LPIPS measures whether matched physical states produce consistent observations. PRDC density and coverage measure whether generated frames remain on the support of real CarRacing observations rather than collapsing to a narrow set of plausible but uninformative track appearances. Full frame-wise and temporal rollout metrics are reported in Table 6.

Metric	MDN-RNN	Transformer	DreamerV3	LeWM	SWM	Rel. Imp. (%)
State-match LPIPS ↓	0.302 ± 0.114	0.198 ± 0.145	0.261 ± 0.098	0.236 ± 0.127	**0.028 ± 0.017**	**+86%**
PRDC Density ↑	0.0684	0.0543	0.0528	0.0055	**0.2101**	**+207%**
PRDC Coverage ↑	0.0325	0.0390	0.1060	0.0060	**0.1230**	**+16%**

The auxiliary diagnostics in Table 3 support the policy-facing interpretation. SWM improves State-match LPIPS, indicating a more stable link between physical state and rendered track appearance, while higher PRDC density and coverage suggest that this agreement is not achieved by image collapse. These results indicate that SWM preserves more controller-relevant track structure than the compact latent-dynamics baselines.

4.4 Horizon diagnostics

The aggregate results above evaluate long rollouts, but they do not by themselves show where errors accumulate. Figure 3 reports frame fidelity as a function of rollout horizon in DMLab and CarRacing. We also include an SWM-oracle variant that uses the same encoded scene code and renderer as SWM, but replaces the autoregressively predicted state with the ground-truth evaluation state before rendering. This removes accumulated transition drift while leaving scene encoding and rendering error unchanged. The flatter oracle curves indicate that much of SWM's remaining degradation is due to state-transition drift rather than loss of scene information inside the renderer. Revisitation-gap curves are reported as supporting diagnostics in Appendix B.1.

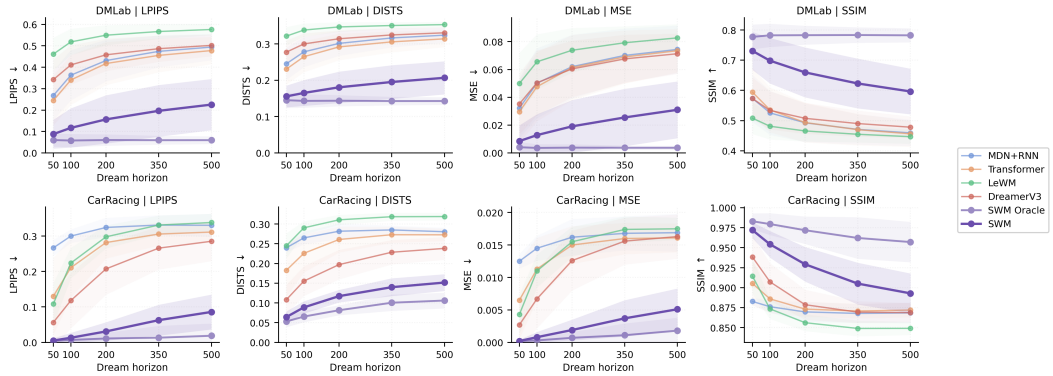


Figure 3: Rollout fidelity versus dream horizon in DMLab and CarRacing. Lower is better for LPIPS, DISTS, and MSE; higher is better for SSIM. SWM degrades more slowly than the latent-dynamics baselines, while SWM-oracle remains flatter, localizing much of the residual long-horizon error to transition drift rather than renderer-side forgetting.

4.5 Matched scene and state information

The entangled-latent control receives the same pretrained map code, initial state, actions, state supervision, data, renderer, sequence length, and training budget as SWM, with comparable recurrent capacity. It folds scene and state into one evolving renderer code, without map reinjection or a separately addressable physical-state component. Neither model receives future states or observations. The transition objectives follow the representations: physical-state deltas for SWM and fused-latent deltas with auxiliary state losses for the control (Appendix B.2).

Table 4: Primary factorization control in DMLab under matched scene and state information. The control reports mean $\pm$ SD over three independent training seeds. Stationary metrics follow 250 exact no-op updates and a 36-view yaw sweep. The control matches short-horizon fidelity but loses place identity with recurrent updates. Full results are in Table 7.

Metric	SWM	Entangled-latent control
$H = 10$ cumulative LPIPS ↓	0.0861	0.0882 ± 0.0003
$H = 200$ cumulative LPIPS ↓	0.1631	0.1581 ± 0.0014
$H = 500$ cumulative LPIPS ↓	0.2256	0.3341 ± 0.0026
State-match LPIPS, gaps 50–499 ↓	0.033	0.3738 ± 0.0236
Out-and-back Recall@1 ↑	0.770	0.260 ± 0.050
Stationary Origin-match LPIPS ↓	0.2615	0.5408 ± 0.0105
Stationary Recall@1 ↑	0.890	0.053 ± 0.015
Stationary yaw error (deg) ↓	32.7	81.2 ± 1.8

Table 4 distinguishes access to scene information from its retention. The control's comparable fidelity through $H = 200$ shows that the shared map code and renderer support accurate short rollouts under both representations. The later divergence emerges as the fused code is repeatedly updated. Stationary retrieval further degrades without any change in the target pose, locating the failure within recurrent prediction rather than actual motion. This can reflect drift of scene or pose information within the fused latent; SWM instead retains a fixed scene reference while updating explicit state. The pattern recurs across three control seeds. We additionally examine sensitivity to scene-code perturbations in Appendix B.3.

4.6 Prior-sampled scene latents

The VAE prior provides a qualitative check on whether SWM can be queried beyond encoded evaluation maps. We sample $z \sim \mathcal{N}(0, I)$ and render observations at selected states. Figure 4 illustrates this generative use of the scene variable but is not used as evidence for spatial persistence; the quantitative claims rely on the diagnostics above.

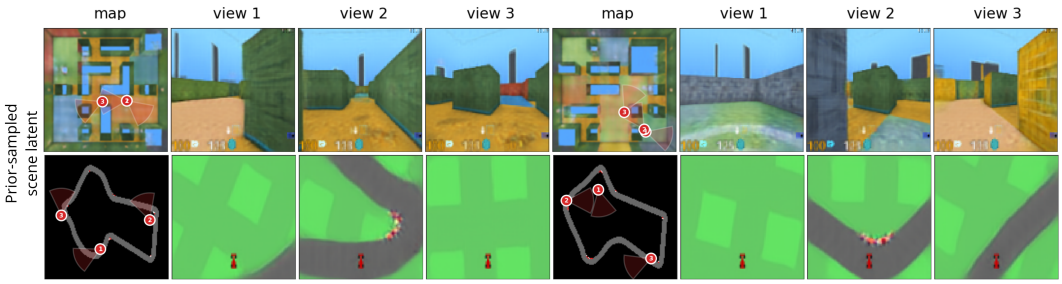


Figure 4: Prior-sampled scene latents in DMLab and CarRacing. We sample $z \sim \mathcal{N}(0, I)$, decode the corresponding map-like representation for visualization, and query the renderer at selected states. Red markers indicate the queried states. This figure is qualitative and is not used as evidence for long-horizon spatial persistence.

5 Discussion, Limitations, and Future Work

The experiments support the paper's central claim: separating scene memory from state evolution improves long-horizon spatial consistency when persistent map-like scene information and low-dimensional state are available. The matched entangled-latent control tests this distinction under equal input information; cross-family comparisons alone cannot establish it. The experiments ask whether structured scene and state information provides an effective inductive bias for coherent imagination under matched rollout data and workstation-scale training budgets. They do not establish superiority for generic video-only prediction, unknown-state settings, changing layouts, or environments without stable map-like observations.

SWM has two main limitations: residual transition drift, suggested by flatter oracle-state curves, and reliance on a compact scene code, static intra-episode geometry, and single-agent low-dimensional state. Providing a map and initial state bypasses mapping and localization, while top-down inputs omit arbitrary side-facing appearance; the method thus does not directly handle unseen, map-free navigation. SRT [27] and NeRF-VAE [26] infer scenes from RGB views, while Semantic MapNet [48] builds allocentric memory from RGB-D and known poses. These are untested candidate front ends, leaving mapping and localization uncertainty unresolved. Larger or dynamic scenes, multiple agents, and policy learning within SWM remain open directions.

Potential benefits include more sample-efficient model-based reinforcement learning, lower-cost simulation, and clearer spatial-coherence diagnostics. The main risk is over-reliance on synthetic rollouts beyond the stated assumptions. All experiments use procedural simulations without human data or deployment.

Acknowledgments and Disclosure of Funding

This work was partially funded by MICIU/AEI/10.13039/501100011033 and FEDER/UE through grants PID2023-147404OB-I00 and PID2021-128045OA-I00; by the Ministerio para la Transformación Digital y de la Función Pública and NextGenerationEU/PRTR through grant TSI-100925-2023-1; CITIC, as a Research Center accredited by the Galician University System, is funded by "Consellería de Cultura, Educación e Universidade from Xunta de Galicia", supported in 80% through the ERDF Operational Program Galicia 2014-2020, with the remaining 20% funded by "Secretaría Xeral de Universidades" (Grant ED431G 2023/01); Xunta de Galicia (Grant ED431C 2026/38) and the European Union ERDF funds.

Competing interests. The authors declare no competing interests.

References

[1] J. Schmidhuber. *Making the world differentiable: on using self supervised fully recurrent neural networks for dynamic reinforcement learning and planning in non-stationary environments*. Forschungsberichte Künstliche Intelligenz. Inst. für Informatik, 1990. URL `https://books.google.es/books?id=9c2sHAAACAAJ`.

[2] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[3] Danijar Hafner, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

[4] Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *The Eleventh International Conference on Learning Representations*, 2023.

[5] Chang Chen, Yi-Fu Wu, Jaesik Yoon, and Sungjin Ahn. Transdreamer: Reinforcement learning with transformer world models. *arXiv preprint arXiv:2202.09481*, 2022.

[6] Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[7] Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[8] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pages 4114–4124. PMLR, 2019.

[9] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[10] Wei-Ning Hsu, Yu Zhang, and James Glass. Unsupervised learning of disentangled and interpretable representations from sequential data. *Advances in neural information processing systems*, 30, 2017.

[11] Yingzhen Li and Stephan Mandt. Disentangled sequential autoencoder. *arXiv preprint arXiv:1803.02991*, 2018.

[12] S. M. Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S. Morcos, Marta Garnelo, Avraham Ruderman, Andrei A. Rusu, Ivo Danihelka, Karol Gregor, David P. Reichert, Lars Buesing, Theophane Weber, Oriol Vinyals, Dan Rosenbaum, Neil Rabinowitz, Helen King, Chloe Hillier, Matt Botvinick, Daan Wierstra, Koray Kavukcuoglu, and Demis Hassabis. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018. doi: 10.1126/science.aar6170. URL `https://www.science.org/doi/abs/10.1126/science.aar6170`.

[13] Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*, 2018.

[14] Konstantinos A Tsintotas, Loukas Bampis, and Antonios Gasteratos. The revisiting problem in simultaneous localization and mapping: A survey on visual loop closure detection. *IEEE Transactions on Intelligent Transportation Systems*, 23(11):19929–19953, 2022.

[15] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[16] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, 2019.

[17] Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021.

[18] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640:647–653, 2025.

[19] Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025.

[20] 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.

[21] 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.

[22] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew J. 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.

[23] Ananya Kumar, SM Eslami, Danilo J Rezende, Marta Garnelo, Fabio Viola, Edward Lockhart, and Murray Shanahan. Consistent generative query networks. *arXiv preprint arXiv:1807.02033*, 2018.

[24] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 405–421, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58452-8.

[25] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems (NeurIPS)*, 33:7537–7547, 2020.

[26] Adam R Kosiorek, Heiko Strathmann, Daniel Zoran, Pol Moreno, Rosalia Schneider, Sona Mokrá, and Danilo Jimenez Rezende. Nerf-vae: A geometry aware 3d scene generative model. In *International conference on machine learning*, pages 5742–5752. PMLR, 2021.

[27] Mehdi SM Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lučić, Daniel Duckworth, Alexey Dosovitskiy, et al. Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6219–6228. IEEE, 2022.

[28] Haian Jin, Hanwen Jiang, Hao Tan, Kai Zhang, Sai Bi, Tianyuan Zhang, Fujun Luan, Noah Snavely, and Zexiang Xu. LVSM: A large view synthesis model with minimal 3D inductive bias. In *International Conference on Learning Representations*, 2025.

[29] Emilio Parisotto and Ruslan Salakhutdinov. Neural map: Structured memory for deep reinforcement learning. *arXiv preprint arXiv:1702.08360*, 2017.

[30] Jingwei Zhang, Lei Tai, Ming Liu, Joschka Boedecker, and Wolfram Burgard. Neural slam: Learning to explore with external memory. *arXiv preprint arXiv:1706.09520*, 2017.

[31] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. *arXiv preprint arXiv:2004.05155*, 2020.

[32] Kuan Fang, Alexander Toshev, Li Fei-Fei, and Silvio Savarese. Scene memory transformer for embodied agents in long-horizon tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 538–547, 2019.

[33] Tong Wu, Shuai Yang, Ryan Po, Yinghao Xu, Ziwei Liu, Dahua Lin, and Gordon Wetzstein. Video world models with long-term spatial memory. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[34] Xiaotao Hu, Mingkai Jia, Xiaoyang Guo, Qian Zhang, Xiao-xiao Long, and Wei Yin. Drivingworld: Constructing world model for autonomous driving via video gpt. In *International Conference on Pattern Recognition*, pages 276–291. Springer, 2026.

[35] Xi Guo, Chenjing Ding, Haoxuan Dou, Xin Zhang, Weixuan Tang, and Wei Wu. Infinitydrive: Breaking time limits in driving world models. *arXiv preprint arXiv:2412.01522*, 2024.

[36] Zeqi Xiao, Yushi Lan, Yifan Zhou, Wenqi Ouyang, Shuai Yang, Yanhong Zeng, and Xingang Pan. Worldmem: Long-term consistent world simulation with memory. *arXiv preprint arXiv:2504.12369*, 2025.

[37] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

[38] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International conference on machine learning*, pages 5301–5310. PMLR, 2019.

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

[40] Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

[41] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

[42] Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. In *International conference on machine learning*, pages 7176–7185. PMLR, 2020.

[43] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[44] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[45] Karl Pearson. Vii. mathematical contributions to the theory of evolution.—iii. regression, heredity, and panmixia. *Philosophical Transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, (187):253–318, 1896.

[46] Leonid V Kantorovich. Mathematical methods of organizing and planning production. *Management science*, 6(4):366–422, 1960.

[47] Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991.

[48] Vincent Cartillier, Zhile Ren, Neha Jain, Stefan Lee, Irfan Essa, and Dhruv Batra. Semantic mapnet: Building allocentric semantic maps and representations from egocentric views. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 964–972, 2021.

[49] Bruno Corcuera, Carlos Eiras-Franco, and Brais Cancela. Unsupervised dynamic feature selection for robust latent spaces in vision tasks. *arXiv preprint arXiv:2510.01758*, 2025.

[50] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[51] Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines. In *International Conference on Learning Representations*, 2025.

[52] Hongkai Zhao. A fast sweeping method for eikonal equations. *Mathematics of computation*, 74 (250):603–627, 2005.

[53] James A Sethian et al. *Level set methods and fast marching methods*, volume 98. Cambridge Cambridge UP, 1999.

A Algebraic properties of additive scene-state fusion

This appendix states the simple algebraic properties used in the main text. They are not intended as a general theory of world models. They apply under the assumptions used by SWM: a deterministic renderer, a scene code fixed within an episode, and additive fusion with an identity pathway for z.

Proposition 1 (Fusion-space error decomposition)**.** *Let $f(z, s) = \alpha\, z + (1-\alpha)\pi_s(s)$. Let $z^\star$ and $s_t^\star$ denote reference scene and state variables, and let $\hat{z}$ and $\hat{s}_t$ denote estimated or predicted variables. Then*

$$f(\hat{z}, \hat{s}_t) - f(z^\star, s_t^\star) = \underbrace{\alpha(\hat{z} - z^\star)}_{\textit{scene-code error}} + \underbrace{(1-\alpha)\big(\pi_s(\hat{s}_t) - \pi_s(s_t^\star)\big)}_{\textit{state error}}. \tag{5}$$

Consequently,

$$\|f(\hat{z}, \hat{s}_t) - f(z^\star, s_t^\star)\|_2 \le \alpha\|\hat{z} - z^\star\|_2 + (1-\alpha)\|\pi_s(\hat{s}_t) - \pi_s(s_t^\star)\|_2. \tag{6}$$

Proof. Substitute the definition of f for both argument pairs and subtract. The norm bound follows from the triangle inequality. □

The decomposition is in fusion space before the nonlinear decoder. It does not imply that pixel-space error decomposes additively. Its diagnostic value is that the scene-code term is fixed within an episode, whereas the state term can accumulate over rollout time. This is the distinction probed by the oracle-state analysis.

Proposition 2 (Closed-form compensation in fusion space)**.** *For any scene code z_A, states s_A, s_B, and $\alpha > 0$, define*

$$z_{\mathrm{cf}} = z_A + \frac{(1-\alpha)}{\alpha}\big(\pi_s(s_A) - \pi_s(s_B)\big). \tag{7}$$

Then $f(z_{\mathrm{cf}}, s_B) = f(z_A, s_A)$. If $G(z, s) = \mathcal{D}_\theta(f(z, s))$ with deterministic decoder $\mathcal{D}_\theta$, then $G(z_{\mathrm{cf}}, s_B) = G(z_A, s_A)$.

Proof. Substituting the definition of z_{cf} gives

$$\begin{aligned} f(z_{\mathrm{cf}}, s_B) &= \alpha\, z_{\mathrm{cf}} + (1-\alpha)\,\pi_s(s_B) && (8)\\ &= \alpha\, z_A + (1-\alpha)\,\pi_s(s_B) + (1-\alpha)\big(\pi_s(s_A) - \pi_s(s_B)\big) && (9)\\ &= \alpha\, z_A + (1-\alpha)\,\pi_s(s_A) = f(z_A, s_A). && (10) \end{aligned}$$

Applying the deterministic decoder to both sides gives the result. □

The compensated code is an algebraic fusion-space construction. It need not correspond to an encoded map or remain on the manifold of valid scenes. The scene-code realization experiment therefore replaces this algebraic target with a nearby encoded scene code and tests whether the continuity target can be approximated by a valid scene representation.

B Extended evaluation details and diagnostics

This appendix provides the audit trail for experimental claims in Section 4. The main text focuses on diagnostics tied to spatial persistence: long-horizon coherence, matched-state revisitation, return-to-origin panorama consistency, and horizon-resolved degradation. Here, we report the full metric suite, define the panorama and geometry diagnostics, provide qualitative rollouts, inspect trajectory drift, and summarize implementation and resource measurements. These results are intended to make the main claims easier to inspect and reproduce, rather than to introduce a separate hierarchy of contributions.

B.1 Extended rollout metrics across domains

Tables 5 and 6 report the complete metric suite for DMLab and CarRacing. Metrics probe different failure modes and should not be collapsed into a single score. Frame-wise metrics measure agreement with logged futures, temporal metrics measure adjacent-frame smoothness, PRDC measures distributional support, ECS measures known-pose geometric consistency in DMLab, and State-match LPIPS tests whether revisited physical states produce stable observations. The latter three are most closely aligned with the paper's central claim, since locally smooth predictions can still fail to preserve place identity.

Let $\{x_i\}_{i=1}^{n_r}$ and $\{y_j\}_{j=1}^{n_g}$ be real and generated feature vectors, and r_i the distance from x_i to its kth nearest other real sample. PRDC density and coverage [42] are

$$\text{Density} = \frac{1}{kn_g} \sum_{j=1}^{n_g} \sum_{i=1}^{n_r} \mathbf{1}[\|y_j - x_i\|_2 < r_i],$$

$$\text{Coverage} = \frac{1}{n_r} \sum_{i=1}^{n_r} \mathbf{1}[\min_j \|y_j - x_i\|_2 < r_i].$$

Density counts overlapping real neighbourhoods containing generated samples; coverage counts real neighbourhoods reached by at least one generated sample. For SIFT correspondences, let e_ℓ be the squared Sampson error [44] under the fundamental matrix induced by known camera motion and calibration. ECS Inlier Ratio is $M^{-1} \sum_{\ell=1}^{M} \mathbf{1}[e_\ell < \tau]$, where M is the number of matches and τ the evaluator's inlier threshold.

Table 5: Full DMLab rollout metrics on held-out imagined rollouts. The final column reports the relative improvement of SWM over the best non-SWM baseline for each metric. For Real vs Pred Ratio, improvement is measured as reduced distance to 1, where 1 indicates variation between real and generated sequences. Negative values indicate that SWM is worse than the best baseline on that diagnostic. Temporal smoothness metrics are included for completeness but are not used as headline evidence, since smooth predictions can still fail to preserve place identity.

Metric	MDN-RNN	Transformer	DreamerV3	LeWM	SWM	Rel. Imp. (%)
LPIPS (mean) ↓	0.4954 ± 0.0619	0.4734 ± 0.0657	0.5766 ± 0.0261	0.5022 ± 0.0484	**0.2256 ± 0.1195**	+52%
DISTS (mean) ↓	0.3247 ± 0.0192	0.3129 ± 0.0240	0.3534 ± 0.0073	0.3318 ± 0.0156	**0.2068 ± 0.0455**	+34%
MSE (mean) ↓	0.0735 ± 0.0163	0.0721 ± 0.0169	0.0823 ± 0.0097	0.0720 ± 0.0141	**0.0310 ± 0.0204**	+57%
SSIM (mean) ↑	0.4618 ± 0.0418	0.4603 ± 0.0438	0.4522 ± 0.0229	0.4766 ± 0.0367	**0.5964 ± 0.0755**	+25%
Real vs Pred Ratio ∼	1.9217 ± 0.5314	1.5912 ± 0.3803	1.7230 ± 0.3725	1.4719 ± 0.2710	**1.1690 ± 0.1138**	+64%
tLPIPS (mean) ↓	**0.1424 ± 0.0216**	0.1637 ± 0.0197	0.1432 ± 0.0165	0.1676 ± 0.0162	0.1789 ± 0.0166	-26%
VMAF (mean) ↑	60.9668 ± 13.1072	**61.8789 ± 14.3868**	32.5066 ± 8.4842	46.2090 ± 11.1573	58.6043 ± 14.0874	-5%
PRDC Precision ↑	0.1904	0.3596	0.0236	0.0950	**0.6240**	+74%
PRDC Recall ↑	0.1426	0.1590	0.0004	0.0308	**0.4668**	+194%
PRDC Density ↑	0.0659	0.1462	0.0057	0.0253	**0.3710**	+154%
PRDC Coverage ↑	0.1024	0.2078	0.0086	0.0402	**0.5288**	+154%
ECS Inlier Ratio ↑	0.2176	0.2396	0.0521	0.2878	**0.3507**	+22%
ECS Sampson Error ↓	1027.0440	955.2452	2784.6178	703.9472	**682.7594**	+3%
ECS Num Matches ↑	12.7513	13.1813	9.0675	10.9129	**14.4699**	+10%
State-match LPIPS ↓	0.208 ± 0.115	0.168 ± 0.140	0.269 ± 0.185	0.407 ± 0.142	**0.033 ± 0.061**	+80%

CarRacing is visually simpler than DMLab but dynamically less forgiving: small state errors can compound through steering, throttle, braking, and inertia. For this reason, we interpret CarRacing primarily through matched-state consistency, policy-facing agreement, and horizon-resolved degradation rather than a single aggregate image metric.

Table 6: Full CarRacing rollout metrics on held-out dreamed trajectories. The final column reports the relative improvement of SWM over the best non-SWM baseline for each metric. CarRacing has simpler visual statistics than DMLab but stronger compounding effects from control and vehicle dynamics, so the most relevant diagnostics are matched-state consistency, distributional support, and horizon-resolved degradation. Temporal smoothness metrics are reported as secondary diagnostics because they may reward stable but less faithful rollouts.

Metric	MDN-RNN	Transformer	DreamerV3	LeWM	SWM	Rel. Imp. (%)
LPIPS (mean) ↓	0.3429 ± 0.0944	0.3235 ± 0.0959	0.2830 ± 0.1096	0.3501 ± 0.0909	**0.1028 ± 0.1320**	+64%
DISTS (mean) ↓	0.2883 ± 0.0484	0.2783 ± 0.0525	0.2407 ± 0.0643	0.3260 ± 0.0509	**0.1636 ± 0.0790**	+32%
MSE (mean) ↓	0.0243 ± 0.0476	0.0237 ± 0.0471	0.0222 ± 0.0455	0.0246 ± 0.0439	**0.0123 ± 0.0449**	+45%
SSIM (mean) ↑	0.8473 ± 0.1396	0.8501 ± 0.1397	0.8504 ± 0.1401	0.8275 ± 0.1362	**0.8710 ± 0.1449**	+2%
Real vs Pred Ratio ∼	2.7438 ± 2.0879	3.1930 ± 1.6908	2.0803 ± 0.4951	2.6904 ± 1.2631	**1.3176 ± 0.7060**	+71%
tLPIPS (mean) ↓	0.0115 ± 0.0080	**0.0051 ± 0.0023**	0.0093 ± 0.0040	0.0100 ± 0.0029	0.0076 ± 0.0031	-49%
VMAF (mean) ↑	90.6517 ± 1.5823	90.5760 ± 1.5614	96.7720 ± 2.4749	**99.8000 ± 2.2672**	96.2227 ± 1.9618	-4%
PRDC Precision ↑	0.1900	0.1726	0.1963	0.0174	**0.4511**	+130%
PRDC Recall ↑	0.1411	0.1274	0.1684	0.0105	**0.1911**	+13%
PRDC Density ↑	0.0684	0.0543	0.0528	0.0055	**0.2101**	+207%
PRDC Coverage ↑	0.0325	0.0390	0.1060	0.0060	**0.1230**	+16%
State-match LPIPS ↓	0.302 ± 0.114	0.198 ± 0.145	0.261 ± 0.098	0.236 ± 0.127	**0.028 ± 0.017**	+86%

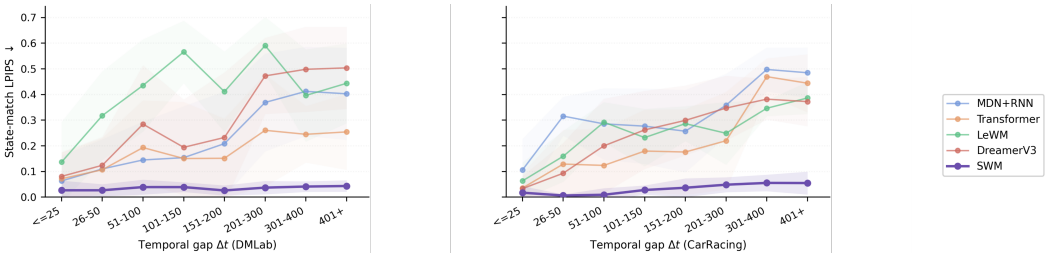


Figure 5: Matched-state LPIPS as a function of the temporal gap between revisits in held-out DMLab and CarRacing rollouts. Lower is better. SWM remains more stable across gap bins than the latent-dynamics baselines, indicating that its consistency advantage is not limited to short-horizon local plausibility.

Temporal smoothness metrics require particular care. Low tLPIPS or high VMAF can favor overly smooth rollouts, including predictions that are visually stable but semantically weak. For the paper's central hypothesis, the most informative rows are the place-sensitive, distributional, and geometry-sensitive diagnostics, together with the horizon-resolved curves in Section 4.4.

Figure 5 reports State-match LPIPS as a function of the temporal gap between matched revisits. These curves support the main revisitation results by showing that SWM remains more stable across temporal gap bins than the latent-dynamics baselines. We keep this figure in the appendix because the main text already reports the aggregate revisitation metrics and the horizon-resolved oracle analysis.

B.2 Matched entangled-latent control

This control tests whether the persistence advantage remains when scene and state information are matched. Both models receive the same pretrained episode map code and initial physical state, actions, state supervision, data, renderer, sequence length, and training budget, with comparable model capacity.

At initialization, the control forms a 256-dimensional fused renderer code $c_0 = \mathrm{encode_fused}(z, s_0)$. A one-layer, 768-unit LSTM predicts increments $c_{t+1} = c_t + \Delta c_t$, and frames are rendered with $\mathrm{decode_fused}(c_t)$. Its complete recurrent state contains c_t and the LSTM hidden and cell states, but no separately addressable map or physical-state component. There is no map reinjection, future-state input, endpoint-state input, or recurrent reset during evaluation. SWM predicts physical-state deltas, while the control predicts fused-latent deltas with auxiliary state losses; the objectives therefore follow the chosen representations.

Table 7: Full DMLab matched-control results. Each control entry is the mean ± SD of the evaluation metric across three independently trained seeds. Cumulative metrics average over the rollout through horizon H, while endpoint LPIPS evaluates the final frame. The stationary regime contains 250 no-op updates at a fixed ground-truth pose before the yaw sweep.

Metric	SWM	Entangled-latent control
$H = 10$ cumulative LPIPS ↓	0.0861	0.0882 ± 0.0003
$H = 200$ cumulative LPIPS ↓	0.1631	0.1581 ± 0.0014
$H = 350$ cumulative LPIPS ↓	0.2136	0.2545 ± 0.0035
$H = 500$ cumulative LPIPS ↓	0.2256	0.3341 ± 0.0026
$H = 500$ endpoint LPIPS ↓	0.3444	0.5553 ± 0.0065
State-match LPIPS, gaps 50–499 ↓	0.033	0.3738 ± 0.0236
Out-and-back Origin-match LPIPS ↓	0.298	0.4713 ± 0.0259
Out-and-back Recall@1 ↑	0.770	0.260 ± 0.050
Stationary Origin-match LPIPS ↓	0.2615	0.5408 ± 0.0105
Stationary Recall@1 ↑	0.890	0.053 ± 0.015
Stationary yaw error (deg) ↓	32.7	81.2 ± 1.8

The control achieves comparable fidelity at short and intermediate horizons but degrades under repeated latent updates. The stationary regime shows a similar failure without physical motion: panorama fidelity and place retrieval deteriorate despite a fixed ground-truth pose. Since the retrieval descriptor is invariant to circular yaw shifts, this cannot be explained by orientation drift alone. However, the control's internal pose is not fixed, so the test cannot distinguish scene forgetting from pose drift within the fused code. SWM instead preserves the scene code while updating explicit state. Together with the oracle-state analysis, these results support isolating scene conditioning from recurrent prediction error. This conclusion is limited to the tested architectures and representation-dependent objectives; the control uses three training seeds, whereas SWM uses one checkpoint.

B.3 Scene-code corruption

On 100 held-out out-and-back episodes, we use three paired draws per episode of $z_\lambda = \mu_\phi(\mathbf{m}) + \lambda\, \sigma_\phi(\mathbf{m}) \odot \eta$, with $\eta \sim \mathcal{N}(0, I)$ and posterior standard deviation σ_ϕ. Table 8 compares noise scales and a wrong-map condition.

Table 8: Sensitivity to scene-code perturbations in DMLab. Values aggregate three paired inference draws per episode.

Scene code	Origin-match LPIPS ↓	Recall@1 ↑
Posterior mean ($\lambda = 0$)	0.3564	0.750
Native posterior scale ($\lambda = 1$)	0.3491	0.760
Strong corruption ($\lambda = 4$)	0.3898	0.477
Wrong map	0.5580	0.000

Native posterior-scale noise gives a paired LPIPS change of -0.0073 (95% CI $[-0.0164, 0.0018]$), with no detectable degradation. Stronger corruption and a wrong map reduce retrieval. This tests local code uncertainty, not robustness to arbitrary mapping errors.

B.4 Panorama consistency under motion and stationary controls

The return-to-origin panorama benchmark tests whether a world model preserves place identity after a long imagined interval. Since single frames can be locally plausible while disagreeing about the surrounding layout, we evaluate 360° panoramas from matched physical poses.

For each held-out episode i, we render an ordered sweep with $V = 36$ yaw bins, separated by 10°. Let

$$P_i^{(0)} = \{I_{i,v}^{(0)}\}_{v=0}^{V-1}, \qquad P_i^{(1)} = \{I_{i,v}^{(1)}\}_{v=0}^{V-1}$$

denote the real start and return panoramas, and let $\hat{P}_i^{(0)}$ and $\hat{P}_i^{(1)}$ denote the corresponding predictions. We use three headline metrics.

First, *Origin-match pano LPIPS* measures whether the model still renders the original place after the return interval. It compares the predicted return panorama to the real start panorama at the same yaw bins:

$$\mathcal{L}_{\text{origin}}(i) = \frac{1}{V} \sum_{v=0}^{V-1} \text{LPIPS}\Big(\hat{I}_{i,v}^{(1)}, I_{i,v}^{(0)}\Big).$$

This strict score does not allow circular yaw correction, so it penalizes place drift, heading drift, and poor rendering. The diagnostics below help separate these cases.

Second, *Origin pano Recall@1* turns the same return-memory question into a retrieval test. Let $b(I)$ denote the LPIPS-backbone frame embedding used by the evaluator, and define the normalized frame embedding

$$e(I) = \frac{b(I)}{\|b(I)\|_2}.$$

The panorama descriptor is the mean of the normalized frame embeddings across yaw bins:

$$\phi(P) = \frac{1}{V} \sum_{v=0}^{V-1} e(I_v).$$

The predicted return panorama is used as a query against the gallery of real start panoramas from all held-out episodes:

$$\text{R@1} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\left[\arg\min_j d\Big(\phi(\hat{P}_i^{(1)}), \phi(P_j^{(0)})\Big) = i\right],$$

where d is the evaluator's descriptor distance. Recall@1 is one only when the generated return panorama retrieves its own real start panorama. This makes the metric a global place-identity test rather than a measure of generic visual plausibility. A model that renders an average-looking corridor can obtain moderate frame quality but low Recall@1 if the generated panorama does not identify the correct episode.

Third, *Pano yaw error* estimates orientation drift by searching over circular shifts between the predicted and real return panoramas. For episode i, define the pairwise perceptual distance matrix

$$M_{uv}^{(i)} = \text{LPIPS}\Big(\hat{I}_{i,u}^{(1)}, I_{i,v}^{(1)}\Big).$$

Each candidate yaw shift $\delta \in \{0, \ldots, V-1\}$ is scored by

$$s_i(\delta) = \frac{1}{V} \sum_{u=0}^{V-1} M_{u,(u+\delta) \bmod V}^{(i)}.$$

The best shift is $\delta_i^\star = \arg\min_\delta s_i(\delta)$, and the reported angular error is

$$\text{YawErr}(i) = \min(\delta_i^\star, V - \delta_i^\star)\frac{360}{V} \quad \text{degrees}.$$

When place identity is retained, this metric identifies a remembered but rotated panorama. It should be interpreted together with Origin-match pano LPIPS and Recall@1, since visually repetitive environments can make multiple yaw shifts partially plausible. A small yaw error alone does not establish persistence when retrieval fails.

Figure 6 reports four additional diagnostics that make the headline metrics easier to interpret.

Start-pose pano LPIPS measures panorama rendering error before long-horizon rollout:

$$\mathcal{L}_{\text{start}}(i) = \frac{1}{V} \sum_{v=0}^{V-1} \text{LPIPS}\Big(\hat{I}_{i,v}^{(0)}, I_{i,v}^{(0)}\Big).$$

This is the panorama-level rendering floor. If this number is already high, later return errors cannot be attributed only to memory or dynamics.

Return-pose pano LPIPS measures fidelity at the final physical pose:

$$\mathcal{L}_{\text{return}}(i) = \frac{1}{V} \sum_{v=0}^{V-1} \text{LPIPS}\Big(\hat{I}_{i,v}^{(1)}, I_{i,v}^{(1)}\Big).$$

This differs from Origin-match pano LPIPS. Return-pose LPIPS asks whether the model matches the real panorama at the final pose. Origin-match pano LPIPS asks whether the return panorama still corresponds to the original place. When the return pose is near, but not exactly equal to, the start pose, these two comparisons test different aspects of consistency.

Pano memory penalty subtracts the initial rendering floor from the return-to-origin error:

$$\Delta_{\text{mem}}(i) = \mathcal{L}_{\text{origin}}(i) - \mathcal{L}_{\text{start}}(i).$$

A value near zero indicates that the long rollout introduces little additional place-matching error beyond the model's initial panorama rendering error. Large positive values indicate additional degradation over the interval. Negative values can occur when the return panorama happens to match the origin better than the initial predicted panorama, so the metric should be interpreted as a difference diagnostic rather than an absolute fidelity score.

Pano structure error tests whether the internal organization of the 360° sweep is preserved. For a panorama P, let $e_v = e(I_v)$ be the normalized LPIPS-backbone embedding of yaw bin v, and define the within-panorama cosine-distance matrix $D(P)_{uv} = 1 - e_u^\top e_v$. The structure error compares predicted and real return panoramas after circular alignment:

$$\mathcal{E}_{\text{struct}}(i) = \min_{\delta} \frac{1}{V(V-1)} \sum_{u=0}^{V-1} \sum_{\substack{v=0 \\ v \neq u}}^{V-1} \Big| D(\hat{P}_i^{(1)})_{uv} - D(P_i^{(1)})_{(u+\delta) \bmod V, (v+\delta) \bmod V} \Big|.$$

The diagonal is excluded because $D(P)_{uu} = 0$ by construction and would otherwise dilute the score. This diagnostic catches panorama collapse and repetitive-view failures: even if individual views look plausible, the full sweep should preserve the same pattern of similarities and differences across directions.

Earlier auxiliary metrics, including margin scores, yaw-aligned LPIPS, and per-view heading retrieval, are not used as headline evidence because they are largely redundant with the three metrics above and are less direct to interpret.

We evaluate these metrics in two regimes. In the out-and-back regime, the model rolls out along controls that move away from the start and return near it, stressing both action-conditioned state prediction and scene persistence. In the stationary-control regime, 250 exact no-op actions leave the ground-truth position and heading fixed, followed by the same in-place 36-view yaw sweep. Recurrent updates remain active throughout; there is no actual translation or rotation during the no-op interval.

When a *real* row is shown, it should be read as a reference rather than as a competing model. Its Origin-match pano LPIPS is not forced to zero, because the return pose is near the start pose but not necessarily identical to it. Its Pano yaw error is exactly zero under the evaluator, which provides a sanity check that the circular-shift diagnostic is behaving as expected.

The two regimes expose different failure modes. Strong stationary-control performance with weaker out-and-back performance indicates accumulated transition drift. Weak performance in both regimes indicates that the model fails to maintain a stable scene-conditioned renderer even without substantial path integration. Poor Start-pose pano LPIPS indicates a rendering or scene-encoding problem before the rollout. In Figure 6, SWM is stronger in the stationary-control regime than in the out-and-back regime, consistent with residual error being dominated by accumulated state drift.

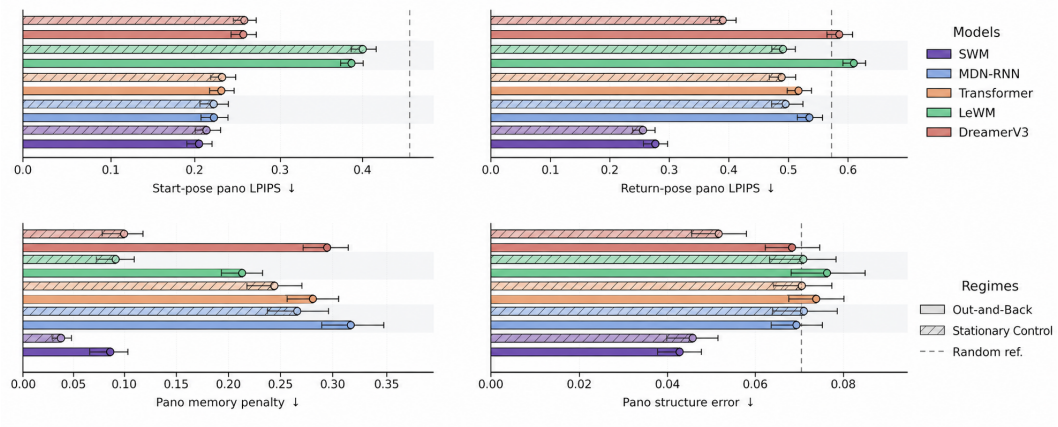


Figure 6: Diagnostic panorama metrics for the out-and-back and stationary-control regimes. Start-pose pano LPIPS measures initial rendering fidelity, return-pose pano LPIPS measures exact fidelity at the return pose, pano memory penalty measures additional place-matching error accumulated during the rollout, and pano structure error measures whether the internal 360° view relationships remain consistent. Lower is better for all four diagnostics.

Representative out-and-back and stationary-control panoramas are shown in Figures 7 and 8, respectively.

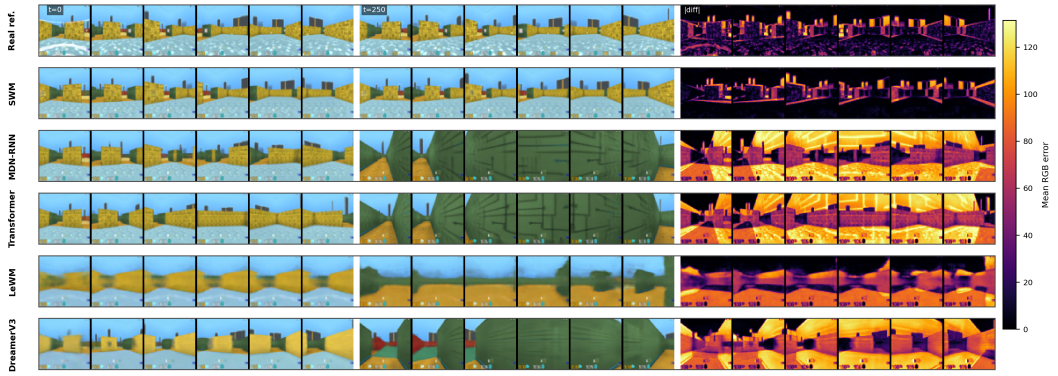


Figure 7: Qualitative out-and-back return panorama on a held-out DMLab episode. SWM preserves the surrounding corridor layout and place identity more faithfully after the excursion, while latent-dynamics baselines show stronger scene drift or panorama inconsistency.

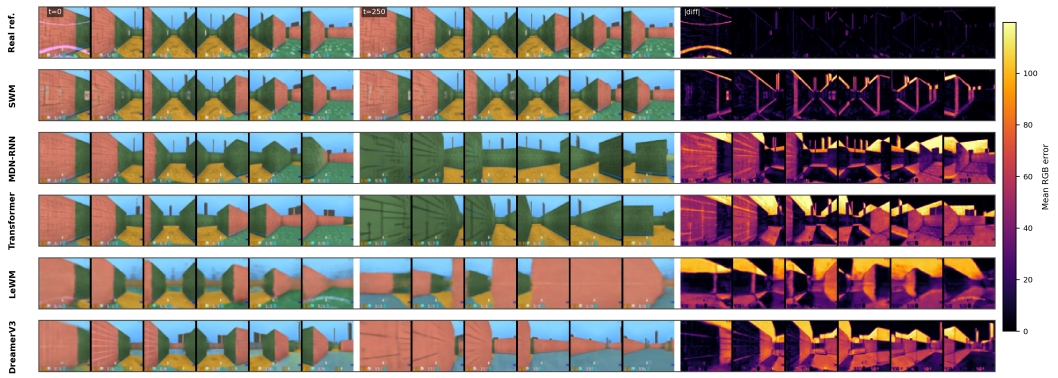


Figure 8: Qualitative stationary-control panorama on a held-out DMLab episode. Errors mainly reflect renderer or scene-memory instability rather than accumulated path-integration drift.

B.5 Qualitative rollouts

Figures 9 and 10 show representative dreamed rollouts on held-out episodes. These examples are qualitative checks, not primary evidence. They visualize the failure mode quantified by the metrics above: latent-dynamics baselines can remain locally plausible while losing the identity of the surrounding place or track segment, whereas SWM more often preserves the association between state, layout, and appearance over longer horizons.

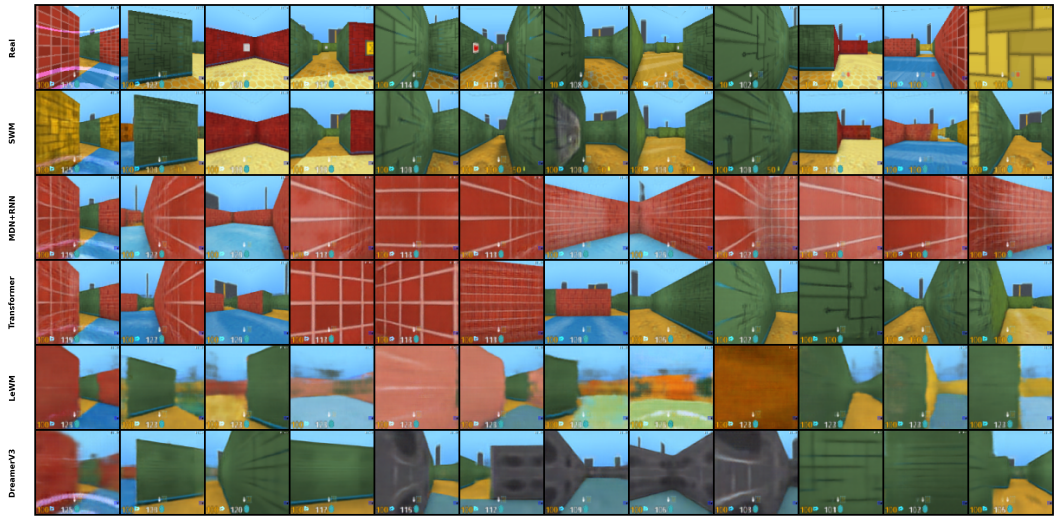


Figure 9: Representative dreamed rollout on a held-out DMLab episode. The comparison illustrates long-horizon place consistency: SWM maintains a more stable association between pose and corridor appearance, while several latent-dynamics baselines drift toward inconsistent views.

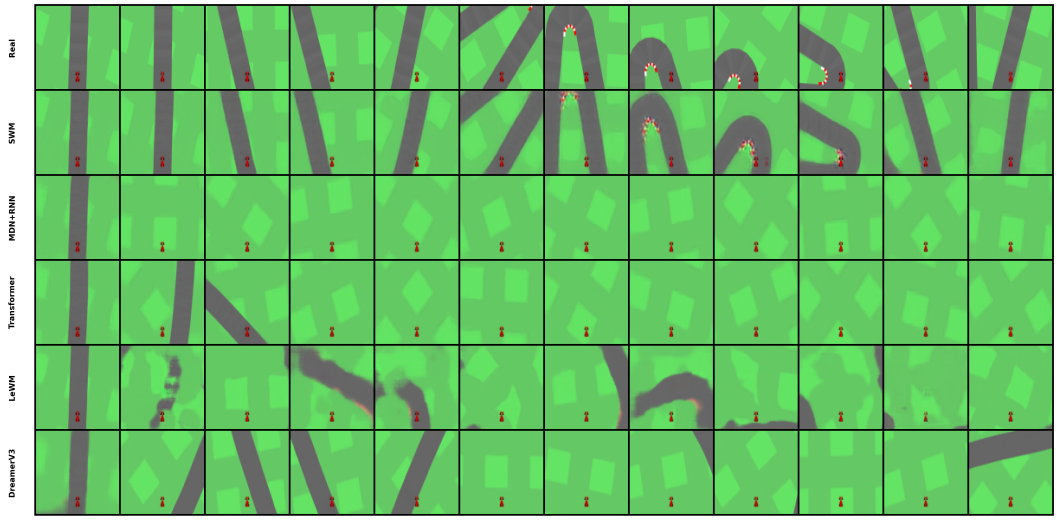


Figure 10: Representative dreamed rollout on a held-out CarRacing episode. While some latent-dynamics baselines drift toward inconsistent views, SWM maintains a more stable association between state and local track appearance.

B.6 Map adherence, collision structure, and trajectory drift

We further inspect whether the predicted dynamics remain grounded in the encoded map. Figure 11 overlays autoregressive predicted trajectories with simulator trajectories on held-out DMLab episodes. The typical SWM failure mode is gradual state drift within a coherent map, rather than abrupt collapse into an unconstrained latent trajectory.

Figure 12 decomposes transition residuals. The residual distribution remains centered around small delta-state errors, while the residual norm increases over rollout time. This supports the interpretation used in the main text and horizon curves: SWM does not eliminate autoregressive error, but it makes the dominant error mode interpretable as accumulated mislocalization in explicit state space.

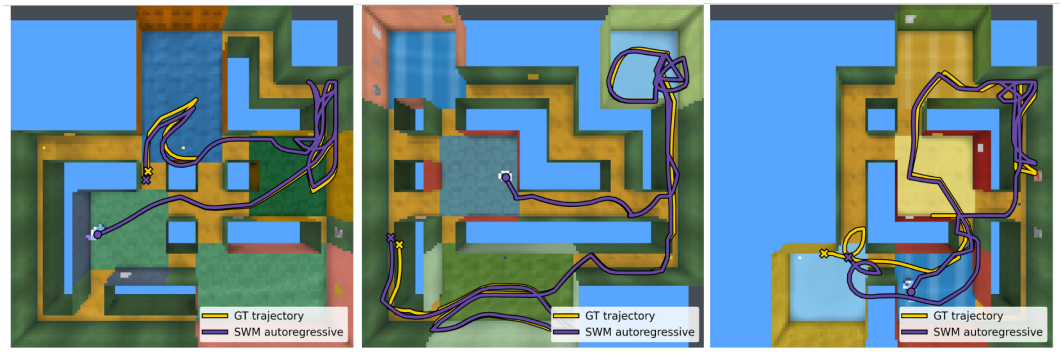


Figure 11: Top-down trajectory overlays on held-out DMLab episodes. SWM generally follows local map structure and obstacle-induced turns, but accumulates gradual autoregressive drift over long horizons.

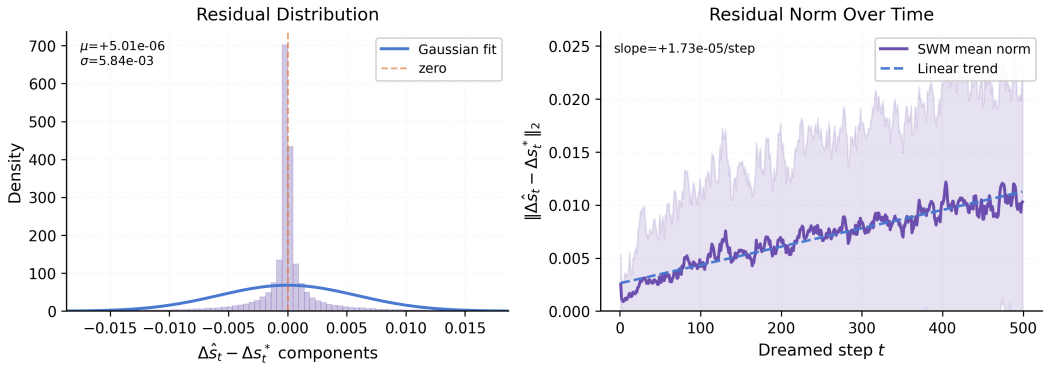


Figure 12: Transition residual analysis for SWM on CarRacing. Left: signed delta-state residuals. Right: residual norm over dreamed time. The main failure mode is gradual accumulation of small state errors rather than immediate dynamical collapse.

B.7 Closed-form compensation with valid scene codes

The closed-form compensation experiment tests whether the additive scene-state interface provides a usable continuity target for renderer-level scene interventions. It asks whether the map-conditioned code can change at a selected timestep without forcing an immediate discontinuity in the current egocentric view.

Given a source scene-state pair (z_A, s_A) and a queried state s_B, Eq. 7 defines z_{cf} such that $f(z_{\mathrm{cf}}, s_B) = f(z_A, s_A)$. Thus, before decoding, the state change is exactly compensated by a shift in the scene-code input. This is a fusion-space target, prior to requiring that the code be realized by the scene model.

We realize the target through the learned scene autoencoder. Let $\mathcal{S}_{\mathrm{dec}}$ denote the map decoder and let $q_\phi(\cdot)$ denote the scene encoder. We form

$$\tilde{\mathbf{m}}_{\mathrm{cf}} = \mathcal{S}_{\mathrm{dec}}(z_{\mathrm{cf}}), \qquad z_B = q_\phi(\tilde{\mathbf{m}}_{\mathrm{cf}}).$$

The rendered sequence is $G(z_B, s_B)$. The code z_B is tied to the closed-form target, while the final scene representation is produced by the same map-conditioned encoder used by SWM. This experiment tests whether the continuity target remains visible after passing through the learned map representation (Figure 13).

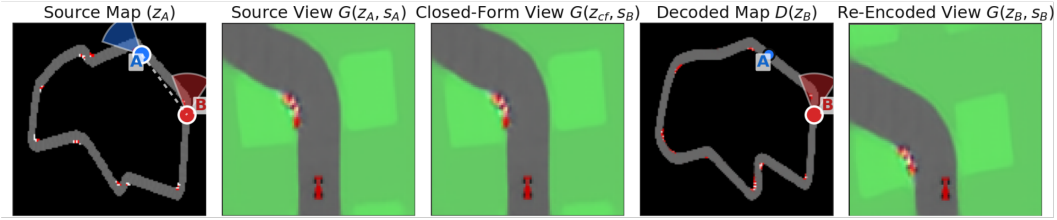


Figure 13: Scene-code realization of closed-form compensation. The source pair (z_A, s_A) defines the view to preserve when the query changes to s_B. The algebraic code z_{cf} exactly matches the fused renderer input but may not correspond to a valid encoded map. We therefore decode z_{cf} through the learned scene decoder and re-encode the resulting map-like representation, producing $z_B = q_\phi(\mathcal{S}_{\mathrm{dec}}(z_{\mathrm{cf}}))$. Rendering $G(z_B, s_B)$ tests whether the fusion-space continuity target remains approximately realizable after projection through the learned scene representation.

C Implementation and reproducibility details

This appendix specifies the SWM instantiation used in the reported experiments. The goal is reproducibility rather than an additional methodological claim. The implementation follows the factorization in Section 3: a scene encoder produces one persistent code per episode, an observation model renders frames from scene code and explicit state, and a transition model predicts action-conditioned state increments.

Notation. Let d_z be the scene-latent dimension, d_s the renderer-state dimension, d_Δ the transition-target dimension, d_a the action dimension, d_f the fusion width, and h_T the transition hidden width. The reported models use $d_z = d_f = 256$ and $L = 10$ Fourier bands for DMLab, and $d_z = d_f = 64$ and $L = 4$ for CarRacing. The transition core uses $h_T = 384$. Decoder-specific widths are given below.

C.1 Scene encoder

We instantiate the scene encoder q_ϕ as a two-stage variational encoder trained on top-down environment maps [37, 49]. Stage 1 learns a deterministic spatial-compression backbone; Stage 2 fits a lightweight variational bottleneck on the frozen Stage 1 representation. This staged design separates map reconstruction from latent regularization, making the scene code useful both for encoding held-out maps and for sampling from the prior.

In Stage 1, the input map $\mathbf{m}$ is processed by a UNet-based preprocessing path [50], a lightweight spatial feature-selection module, and learned downscaling blocks to produce a compact bottleneck $\mathbf{h} \in \mathbb{R}^{16\times16\times16}$. A mirrored upscaling path and refinement UNet reconstruct the input map from $\mathbf{h}$. Stage 1 is trained with reconstruction losses only.

In Stage 2, the Stage 1 backbone is frozen and only the compact VAE bottleneck is trained:

$$\begin{aligned}
&\mathrm{Enc}_{\mathrm{mini}}: \ 16\times16\times16 \to 8\times8\times128 \to 4\times4\times256 \to (\mu, \log\sigma^2) \in \mathbb{R}^{d_z},\\
&\mathrm{Dec}_{\mathrm{mini}}: \ z \to 4\times4\times256 \to 8\times8\times256 \to 16\times16\times128 \to 16\times16\times16.
\end{aligned}$$

C.2 Observation model

The observation model renders an RGB frame $\hat{o}_t \in \mathbb{R}^{96\times96\times3}$ from the persistent scene code z and the current normalized agent state s_t.

State pathway. The state branch maps $s_t \in \mathbb{R}^{d_s}$ into the fusion space by Fourier features followed by a linear projection, normalization, and dropout:

$$s_t \xrightarrow{\gamma_s} \mathbb{R}^{2Ld_s} \to \mathrm{Linear}(2Ld_s, d_f) \to \mathrm{LayerNorm} \to \mathrm{Dropout} = \pi_s(s_t).$$

For DMLab, the renderer state is $(x, y, \sin\theta, \cos\theta)$. For CarRacing, it is $(x, y, \sin\theta, \cos\theta, v_x, v_y, \omega)$. All quantities are normalized with the environment-specific state transform.

Latent pathway and fusion. The scene code enters the renderer through an identity pathway. No mapping network or learned projection is applied to z before fusion. The state pathway maps the low-dimensional state to the same dimensionality through Fourier features followed by a linear projection, yielding $\pi_s(s_t) \in \mathbb{R}^{d_z}$.

The renderer combines the projected state and scene code using the same bounded additive fusion as Eq. 4:

$$f_t = \alpha\, z + (1-\alpha)\pi_s(s_t), \qquad \alpha \in [\epsilon, 1-\epsilon].$$

The coefficient α is a learned global scalar shared across all scenes and episodes within each domain. We set $\epsilon = 0.05$ in both domains without metric-specific tuning. Thus, the coefficient of each pathway is at least 0.05; this does not imply a corresponding fraction of output information or pixel variation. The learned values are 0.390 in DMLab and 0.068 in CarRacing. The latter lies near the lower bound, consistent with similar physical states producing similar images across tracks and making a state-only shortcut attractive.

This bounded fusion has two practical roles. First, it prevents the scene coefficient from vanishing, discouraging reliance only on explicit state-observation correlations. It does not guarantee how

the nonlinear decoder uses either pathway. Second, it preserves the identity-path additive structure used in Appendix A: because z enters the renderer directly, the fusion-space decomposition and the closed-form compensation in Eq. 7 are expressed in the same latent coordinates used by the observation model. The fused vector is LayerNorm-normalized before decoding.

Decoder. The fused vector is mapped to an image by an MLP stem and convolutional decoder:

$$\begin{aligned} f_t \;&\to \mathrm{Linear}(d_f, h_G) \;\to\; \mathrm{RMSNorm} \;\to\; \mathrm{SiLU} \\ &\to \mathrm{ResMLP} \;\to\; \mathrm{Linear}(h_G, C_0 H_0 W_0) \\ &\to \mathrm{reshape}(C_0, H_0, W_0) \;\to\; \mathrm{ConvDecoder} \;\to\; \hat{o}_t. \end{aligned}$$

Each residual MLP block has the form $x + \mathrm{Linear}(\mathrm{SiLU}(\mathrm{RMSNorm}(x)))$. In the reported model, the residual MLP uses hidden width $h_G = 256$, the spatial projection produces a $4 \times 4 \times 512$ feature map, and the convolutional decoder applies residual convolution blocks interleaved with four learned upsampling stages. A final bilinear resize maps the output to 96×96, followed by a three-channel $\tanh$ head. The renderer is trained with $\mathcal{L}_{\mathrm{render}} = 0.8\,\mathcal{L}_{\mathrm{MSE}} + 0.2\,\mathcal{L}_{\mathrm{LPIPS}}$.

Renderer ablation. Table 9 ablates the main renderer components on DMLab validation frames. Fourier state features and perceptual supervision are the largest contributors: replacing Fourier features with a linear state projection increases LPIPS by 35.6%, while removing LPIPS increases it by 156.8%. State dropout gives a smaller improvement in this evaluation. The fusion ablation is the most relevant to the paper's formulation: replacing additive scene-state fusion with concatenation changes LPIPS by only 0.9%. Thus, in this setting, the identity-path additive fusion preserves the closed-form structure used by SWM without a measurable loss in renderer fidelity. The minimal renderer, which combines no Fourier features, concatenation fusion, and no state dropout, degrades LPIPS by 25.9%.

Table 9: Generator ablation on DMLab validation frames. Δ LPIPS is measured relative to the full renderer. Fourier state features and perceptual supervision provide the largest gains. Concatenation fusion replaces the additive identity-path scene-state interface with a learned joint input representation; its similar LPIPS indicates that the additive fusion used for SWM's closed-form structure does not impose a visible fidelity cost in this validation setting.

Variant	LPIPS ↓	SSIM ↑	MSE ↓	Δ LPIPS
Full renderer	**0.0672**	**0.7650**	0.0038	–
Linear state encoding	0.0911	0.6735	0.0063	+35.6%
Concatenation fusion	0.0678	0.7405	**0.0036**	+0.9%
No state dropout	0.0704	0.7491	0.0040	+4.8%
MSE only	0.1725	0.7616	0.0040	+156.8%
Minimal renderer	0.0845	0.6939	0.0055	+25.9%

C.3 Transition model

The transition model predicts additive state increments conditioned on the current state, the persistent scene code, and the action:

$$\Delta s_t = T_\psi(\pi_s(s_t), z, \pi_a(a_t)), \qquad s_{t+1} = s_t + \Delta s_t.$$

The model is trained to predict normalized deltas rather than absolute next states, because local motion is small relative to absolute coordinates and because delta errors are easier to interpret geometrically during rollout.

In rollout, the updated heading angle of the agent in s_{t+1} is converted back to $(\sin\theta, \cos\theta)$ before rendering. This keeps the renderer input continuous while preserving an additive transition target for orientation.

Input branches. The transition state branch uses the same Fourier-feature parameterization as the renderer, but with separate learned weights:

$$s_t \xrightarrow{\gamma_L} \mathbb{R}^{2Ld_s} \to \mathrm{Linear}(2Ld_s, d_f) \to \mathrm{RMSNorm} = \pi_s(s_t).$$

and the action branch is

$$a_t \to \mathrm{Linear}(d_a, d_f/2) \to \mathrm{LayerNorm} \to \mathrm{SiLU} = \pi_a(a_t).$$

Recurrent core and output head. The transition model concatenates projected state features, the raw scene code, and projected action features:

$$[\pi_s(s_t), z, \pi_a(a_t)] \to \mathrm{Linear}(d_f + d_z + d_f/2, h_T) \to \mathrm{LSTM}(h_T, h_T).$$

The output head is

$$h_t \to \mathrm{Linear}(h_T, h_T/2) \to \mathrm{RMSNorm} \to \mathrm{SiLU} \to \mathrm{Linear}(h_T/2, d_\Delta).$$

Predicted deltas are de-normalized before being added to the current state during autoregressive rollout.

Heading update. For rendering, orientation is represented as $(\sin\theta, \cos\theta)$, avoiding discontinuities at angular wrap boundaries. For dynamics, the transition target uses an additive heading increment $\Delta\theta$. During rollout, we recover the current angle with $\mathrm{atan2}$, add the predicted increment, and convert the updated heading back to sine-cosine form. This keeps renderer inputs continuous while making rotational drift explicit.

Transition-core selection. CarRacing is the hardest transition test in our experiments because the model must integrate steering, throttle, brake, inertia, and heading curvature over long rollouts. We therefore compared LSTM, Transformer, xLSTM, and Mamba cores under the same state, action, scene-code inputs, and supervision budget. As shown in Figure 14, the one-layer LSTM produced the most reliable held-out autoregressive trajectories. In this low-dimensional supervised dynamics setting, larger sequence cores did not improve horizon-level trajectory accuracy under our training budget, suggesting that the main bottleneck is local dynamics generalization rather than context length.

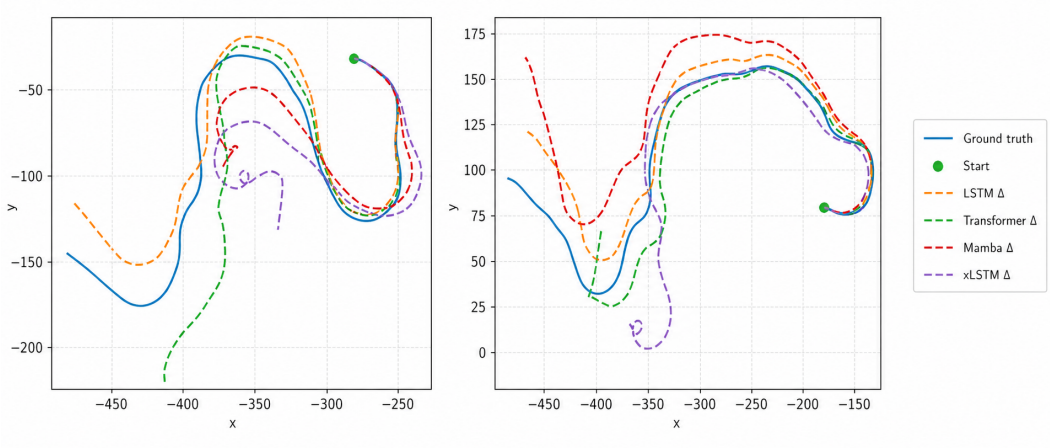


Figure 14: Transition-core comparison on held-out CarRacing rollouts. Each panel overlays autoregressive predicted trajectories against the simulator trajectory under the same action sequence. The one-layer LSTM core generalizes most reliably in this low-dimensional state-action regime. Larger sequence models did not improve long-horizon trajectory accuracy under the same supervision budget.

C.4 Training summary

SWM is trained in three stages. First, the scene encoder is fit on top-down maps, with the deterministic spatial-compression backbone trained before the variational bottleneck. Second, the renderer is trained on tuples (z, s_t, o_t) using precomputed scene codes and the weighted MSE plus LPIPS objective. Third, the transition model is trained with teacher forcing on action-conditioned state sequences to predict normalized state deltas.

We use staged training for diagnostic clarity rather than as a claim of optimality. It keeps scene encoding, rendering, and dynamics separable during evaluation, and it makes the oracle-state analysis in Section 4.4 straightforward to interpret.

C.5 Optimization and baseline configurations

All experiments use a seed of 101, 20,000 episodes of 500 steps, RGB observations of size $96 \times 96 \times 3$, and a $99/1$ train-validation split over episodes. SWM trains its scene encoder on 50,000 top-down maps of $128 \times 128 \times 3$. We report the effective epoch count of the checkpoint used for evaluation and profiling.

Each baseline retains its defining predictive architecture and objective, with image resolution, action dimensionality, and capacity adapted to the common workstation-scale regime. Capacity and checkpoints are not selected on the test set. We monitor held-out validation loss under the learning-rate schedules below and train until this objective plateaus.

SWM. The scene encoder is trained in two stages with AdamW and cosine learning-rate decay. Stage 1 uses a learning rate of 10^{-2} to 10^{-4}, weight decay of 5×10^{-5}, a batch size of 1024, and 31 epochs. Stage 2 freezes Stage 1 and trains the variational bottleneck with a learning rate of 10^{-4} to 10^{-5}, a free-bits threshold of 0.05, a KL weight of $\beta = 0.5$, a batch size of 256, and 117 epochs. The renderer uses AdamW with a learning rate of 5×10^{-4} to 5×10^{-5}, weight decay of 10^{-5} applied only to non-normalization, non-bias, non-scalar parameters, a batch size of 1024, state dropout of 0.1, and a loss of $0.8\,\mathcal{L}_{\mathrm{MSE}} + 0.2\,\mathcal{L}_{\mathrm{LPIPS}}$ for 29 epochs. The transition model uses AdamW with a learning rate of 10^{-4} to 10^{-5}, weight decay of 10^{-5}, a sequence length of 200, a stride of 20, a batch size of 32, and 21 epochs. It is trained with teacher forcing on normalized state deltas and validated with autoregressive rollouts.

Baselines. MDN-RNN and the causal Transformer use a pre-trained pixel-space encoder with 512-dimensional latents, trained with AdamW, a learning rate of 2×10^{-4} to 10^{-6}, weight decay of 10^{-5}, a batch size of 256, a loss of $0.8\,\mathcal{L}_{\mathrm{MSE}} + 0.2\,\mathcal{L}_{\mathrm{LPIPS}}$, and 28 epochs. MDN-RNN uses a hidden size of 512, five Gaussian mixture components, AdamW, a learning rate of 3×10^{-4} to 10^{-5}, weight decay of 10^{-5}, a batch size of 32, a sequence length of 200, a stride of 20, and 61 epochs. The causal Transformer is an 8-layer, 12-head autoregressive Transformer with a width of 768 and a context length of 500, trained with AdamW, a learning rate of 10^{-4} to 5×10^{-6}, gradient clipping at 1.0, no weight decay on normalization or bias parameters, and 135 epochs. DreamerV3 uses a 512-dimensional continuous latent interface and a discrete RSSM with 32 categorical variables of 32 classes, trained with AdamW, a learning rate of 10^{-4} to 10^{-5}, no weight decay, a sequence length of 64, a batch size of 16, free bits of 1.0, a KL-balance coefficient of 0.5, unimix of 0.01, and 28 epochs. LeWM uses a 512-dimensional JEPA embedding, a 12-layer ViT encoder, and an 8-layer predictor, trained with AdamW, 8 warmup epochs followed by learning-rate decay from 5×10^{-5} to 10^{-6}, weight decay of 1×10^{-5}, a batch size of 256, a sequence length of 200, a stride of 20, SIGReg weight of 0.09, and 73 epochs. For pixel-space evaluation, LeWM trains a CNN decoder on frozen embeddings with AdamW, a learning rate of 1.5×10^{-4} to 10^{-6}, weight decay of 10^{-6}, a batch size of 320, sample stride of 2, and 62 epochs. Table 10 reports the combined footprint across stages.

C.6 Resource footprint and throughput

Recent neural simulation work increasingly emphasizes interactive generation at scale [51, 7]. SWM targets a different operating point: a compact factored world model that remains trainable on a single desktop GPU while sustaining high end-to-end rollout throughput. Table 10 summarizes the training footprint of each system. Table 11 reports end-to-end SWM throughput on both GPU and CPU.

Table 10: Training footprint on a single RTX 3090 Ti. For multi-stage systems, parameters, GPU-hours, and training compute are summed across stages; peak memory is the maximum allocated memory observed over stages.

Model	Params (M)	Peak mem. (GB)	GPU-h	Train PFLOPs
SWM VAE	6.83	3.48	0.73	18.47
SWM Generator	12.40	4.29	34.98	1314.51
SWM Transition	4.90	0.34	1.29	5.00
SWM total	24.14	4.29	37.0	1337.98
MDN-RNN [2]	18.30	2.22	21.91	913.52
Transformer [4]	64.40	10.21	34.97	2367.10
DreamerV3 [18]	24.17	6.72	58.19	826.4
LeWM [20]	32.95	2.89	37.24	2025.79

All experiments were run on one RTX 3090 Ti, with an Intel i7-12700K CPU and 64 GB RAM. For multi-stage systems, we sum learned parameters, GPU-hours, and total training compute across stages, and report the maximum peak allocated GPU memory over stages. Total training compute is estimated from measured profiler TFLOP/s and total training time. All models were trained to convergence under their respective stable schedules. All runs used bf16-mixed precision.

The training numbers place SWM in a useful practical regime. It is not the smallest system in total compute, but it remains a single-workstation method and keeps the deployed model compact at 24.14M learned parameters. This matters for the paper's central claim: the gains in long-horizon scene consistency are not obtained by scaling to a massive video model, but by changing the state representation and factorization.

Table 11: End-to-end SWM rollout throughput in DMLab. The benchmark measures the full simulator, not an isolated submodule.

Device	1-env FPS	$\geq$30 FPS: envs / total FPS	$\geq$35 FPS: envs / total FPS
CPU (i7-12700K)	56.1	4 / 140.5	4 / 140.4
GPU (RTX 3090 Ti)	701.1	231 / 6953.2	196 / 6876.4

This makes SWM practical not only as an offline diagnostic model, but also as a lightweight neural rollout model for imagination-based reinforcement learning in domains satisfying the stated assumptions.

C.7 Fast-sweeping-guided data collection

For DMLab, we collect training rollouts without a pretrained exploration policy, using the same top-down maps later encoded by SWM. Each map is segmented into a binary occupancy grid, where free space defines valid agent regions and occupied cells correspond to walls.

Given this grid, we sample a waypoint in free space and compute a geodesic distance field via fast sweeping [52, 53]. The negative spatial gradient of this field induces a locally consistent descent direction, which is mapped to environment actions based on the current agent pose. To avoid nearly deterministic shortest-path trajectories, we inject small stochastic perturbations into the control, improving state-space coverage and observation diversity.

This procedure (Figure 15) is used only as a data-generation mechanism, not as a separate methodological contribution. It provides broad coverage in procedurally generated mazes when pretrained navigators are unavailable, while avoiding the poor coverage and local trapping often produced by naive random policies.

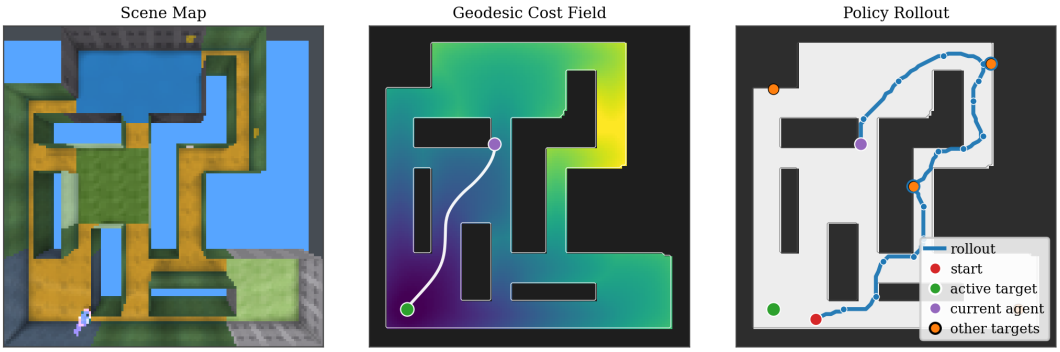


Figure 15: Fast-sweeping-guided rollout generation in DMLab. Left: top-down map of a procedurally generated maze. Middle: geodesic distance field induced by a waypoint on the traversable region; descending its gradient yields a map-consistent control direction. Right: example rollout obtained by iteratively updating waypoints and following the induced field, with stochastic perturbations to avoid deterministic shortest-path trajectories.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction state the scoped contribution: SWM is evaluated in map-conditioned, static-layout navigation-like domains with low-dimensional agent state. The claims are supported by the DMLab and CarRacing experiments and are explicitly limited in Section 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: Section 5 discusses the main limitations, including reliance on top-down scene input, explicit state supervision, static intra-episode geometry, single-agent settings, and residual transition drift.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The theoretical content is limited to algebraic properties of the additive scene-state fusion. Appendix A states the assumptions, gives the propositions, and provides complete proofs for the stated identities.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes].

Justification: The paper provides the main information needed to reproduce the reported experimental results, including the data scale and held-out evaluation protocol in Section 4.1, metric definitions in Appendix B, architecture and training details in Appendix C, baseline configurations in Appendix C.5, and compute resources in Appendix C.6.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes].

   Justification: The anonymized supplementary material includes the compact SWM implementation, dependency specification, env configurations, data-collection commands, training commands, rendering code, metric code. Generated HDF5 datasets and paper-scale checkpoints are not included due to size. The supplement explains how to build DeepMind Lab and regenerate the required artifacts; DeepMind Lab itself is not redistributed because it is an external dependency.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: Section 4.1 specifies the experimental domains, data scale, held-out evaluation setting, rollout length, and main evaluation protocols. Appendix C and Appendix C.5 provide the architecture, state representations, training stages, losses, optimizer settings, batch sizes, sequence lengths, learning-rate schedules, epoch counts, random seed, train-validation split, and baseline configurations.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes].

Justification: All reported quantitative results, including tables and figures, are presented as *mean* $\pm$ *standard deviation* over held-out evaluation episodes, trajectories, or dreamed rollouts. This includes main tables, horizon curves, and bar plots. These uncertainties capture variability induced by environment stochasticity and rollout sampling. We do not report variation over independently trained random seeds.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer:[Yes].

Justification: Appendix C.6 reports the hardware, memory, GPU-hours, estimated PFLOPs, parameter counts, precision setting, and rollout throughput for the reported systems. These details are sufficient to understand the compute footprint of the reported experiments.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes].

Justification: To the best of our knowledge, the research complies with the NeurIPS Code of Ethics. The experiments are conducted in simulation using procedural environments and

standard RL environments, without human subjects, private data, crowdsourcing, surveillance data, or personally identifiable information. The main risks are over-reliance on synthetic rollouts outside the stated assumptions and environmental cost; these are addressed through the scoped limitations and resource-footprint reporting.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes].

    Justification: Section 5 discusses potential positive impacts, including sample-efficient model-based reinforcement learning and lower-cost simulation, as well as the main negative risk of over-reliance on synthetic rollouts outside the stated assumptions.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: The paper does not release data or models with a high risk of misuse, such as scraped datasets, human-data models, large language models, or general-purpose image generators. The experiments are limited to simulation-only procedural environments.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes].

    Justification: The paper cites the external environments, benchmarks, and evaluation methods used in the experiments, including DeepMind Lab, Gym/CarRacing, LPIPS, PRDC, SIFT-based geometry, and related baselines. The supplementary material does not redistribute DeepMind Lab itself; instead it provides a custom level script and instructions for using an independently built DeepMind Lab installation, together with third-party notices and dependency specifications.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes].

    Justification: The supplementary material documents the released SWM code, DMLab env configuration, generated-data schemas, reproduction commands, and qualitative rollout videos. The package includes a manifest, dependency file, third-party notices, and usage instructions; all released assets are anonymized for review and contain no human-subject or private data.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.

- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A].

    Justification: The paper does not involve crowdsourcing, user studies, human-subject experiments, or human data collection. All experiments are conducted in simulation.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: The paper does not involve crowdsourcing or research with human subjects, so IRB approval or equivalent review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A].

    Justification: The core method, experiments, data generation, and evaluation pipeline do not use LLMs as important, original, or non-standard components. Any LLM assistance limited to writing, editing, or formatting does not affect the methodology, scientific rigor, or originality of the research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.