GEOMETRY-CONDITIONED LATENT WORLD MODELS WITH RECURRENT ADAPTATION IN DYNAMIC ENVIRONMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Autonomous navigation frequently relies on pre-mapped 3D spatial priors, but real-world deployments are inherently dynamic, resulting in inconsistencies between static maps and observations. Latent world models offer a promising, lightweight solution operating solely on RGB video, but current architectures lack the explicit spatial understanding and persistent memory to plan effectively in dynamical environments. To bridge this gap, we introduce GeoMamba-WM, a geometry-grounded recurrent world model that encodes a pre-mapped semantic 3D point cloud and utilizes a Mamba-based State Space Model (SSM) to dynamically correct a persistent hidden state using live visual observations. Prediction and planning are performed within a fused latent space derived via cross-attention between the observation and corrected geometry prior and stabilized by an auxiliary objective that encourages information captured by the SSM state to persist throughout the predictive architecture. To systematically evaluate this, we introduce GOI-NAV, a 3D navigation benchmark that features geometry-observation incongruences such as displaced and unseen obstacles, deformable surfaces, and visual noise. Empirical evaluations demonstrate that GeoMamba-WM significantly outperforms baseline latent world models across all incongruence axes, establishing a new standard for offline, reward-free predictive planning. Ultimately, our results show that latent world models are highly viable for planning in dynamic, pre-mapped 3D environments using depth-less RGB input, demonstrating that cross-modal inconsistencies can be resolved entirely in latent space without explicit mapping or online reconstruction objectives.

1 INTRODUCTION

Latent world models are promising for robotic planning because they avoid pixel-space reconstruction and instead predict future observations directly in latent space. This approach enables lightweight architectures that focus on task-relevant visual information. Recent works like DINO-WM (Zhou et al., 2025) use the pretrained DINOv2 visual encoder and its latent space, while jointly-trained models such as PLDM (Sobal et al., 2025) and LeWM (Maes et al., 2026b) learn the encoder along with the predictor, therefore aligning the latent space to best represent information important to the task. However, these models lack dedicated architectural mechanisms for explicitly processing and persistently memorizing spatial information, causing them to fail completely on continuous-control tasks in dynamic 3D environments.

In the real world, candidate environments for autonomous navigation frequently have access to pre-mapped, static, spatial priors. In domains such as factory automation, domestic robots, and digital surgery, these priors are often available in the form of CAD models, digital twins, or anatomical scans such as MRIs. However, these environments are inherently dynamic, and there is a sim-to-real gap during policy execution, so a pre-planned simulation path cannot be relied upon. During deployment, shifted obstacles, deformable objects, or new, unmapped entities further make the pre-mapped geometry incongruent with the real world. Additionally, inconsistencies in IMU sensors such as drift or noise motivate the reliance on RGB video to navigate the dynamic environment. Resolving these inconsistencies typically requires high-fidelity, continuous depth sensing using LiDAR or RGB-D

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

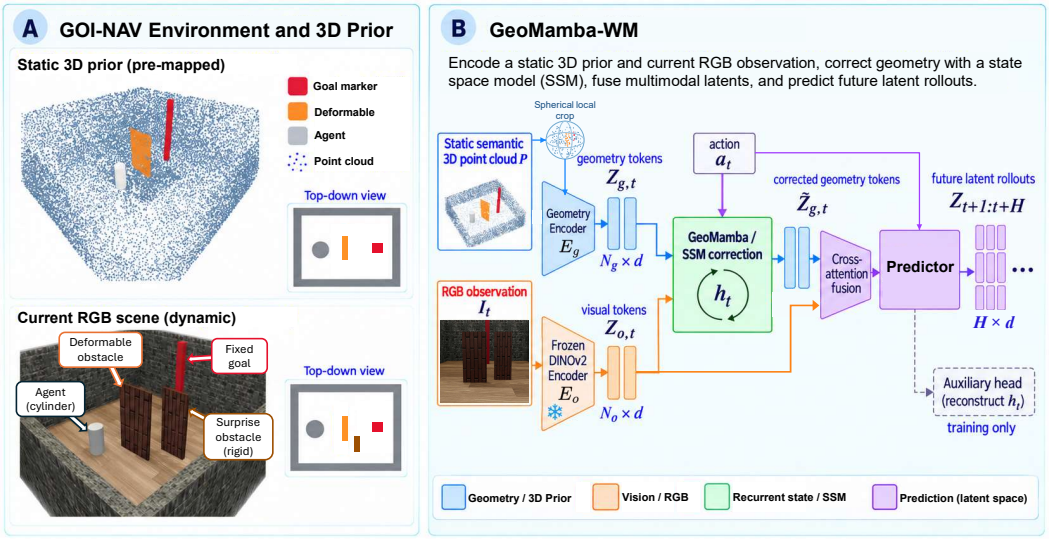

Figure 1: **GeoMamba-WM Overview.** **(A)** A static semantic 3D prior can become inconsistent with the physical environment as obstacles move, appear, or deform. **(B)** Frozen DINOv2 and geometry encoders embed an egocentric visual observation and a local sample of the geometric prior. An observation- and action-conditioned Mamba-based SSM module maintains a persistent hidden state and corrects inconsistent geometry. Cross-attention fuses visual features with the corrected geometry tokens, and an action-conditioned predictor autoregressively generates future fused latents for planning. A training-time auxiliary head predicts the SSM hidden state from encoded and predicted latents, encouraging the fused latent space to learn information from the geometry correction.

cameras. In embodiments where such sensors are not available, the agent must use solely RGB video to navigate this dynamic environment instead. Existing works that aim to solve the 3D navigation problem include SLAM, 3D reconstruction, and most recently, video world models (Garcin et al., 2026) which utilize these prior techniques. These methods, however, involve expensive feature correspondence computation, which further breaks down in visually noisy environments, or dense reconstruction, which is also expensive during both training and planning.

To bridge this gap, we introduce **GeoMamba-WM**, a geometry-grounded latent world model that uses a Mamba block (Gu & Dao, 2024) to dynamically update a static 3D prior. Building upon the traditional latent world model architecture of DINO-WM, we add a pre-trained geometry encoder that samples and encodes a local subset of a static semantic 3D point cloud of the environment. Furthermore, deformability can be learned, and prior deformability knowledge can be introduced via binary semantic deformability labels in the 3D point cloud. The encoded geometry representation is then corrected using a State Space Model (SSM), which uses visual observation embeddings to update a persistent geometric state as the environment evolves and new observations reveal more accurate information. A trainable cross-attention fusion module combines the corrected geometry state with the visual observation into a fused latent representation, which is then used for prediction and planning. To further enforce persistence of geometry information from the SSM to the prediction representation, we add a training-time auxiliary head and objective that reconstructs the SSM hidden state from both encoded and predicted fused latents.

To evaluate our model's capability to navigate in dynamic environments which have a static pre-mapped prior, we introduce **GOI-NAV**, a 3D navigation benchmark designed to test planning performance across several spatial disruptions and dynamic environment variations. Our benchmark tests agents on 3D spatial reasoning tasks where there exists a 3D point cloud of the scene, but also (1) geometric discrepancies ranging from surprise objects to dynamic displacements of objects present in the 3D point cloud, (2) deformable obstacles, and (3) visual noise such as occlusion and noise. These variations demonstrate the general usefulness of a deformability-aware 3D prior and its utilization in GeoMamba-WM, and also demonstrate where static handling causes planning-relevant discrepancies between the 3D model and the current real scene.

Our experiments demonstrate that our SSM-augmented geometry fusion architecture significantly outperforms the latent world model baseline of DINO-WM in both seen and zero-shot incongruent scenarios, enabling real-time, offline, reward-free planning without the significant latency and overhead of explicit pixel or 3D reconstruction. GeoMamba-WM therefore has significant practical value; we show that even a loose 3D picture of a scene can be used to improve latent world models' planning performance. Our main contributions are:

- We introduce **GeoMamba-WM**, a geometry-grounded recurrent world model architecture that uses visual observations to dynamically correct a static semantic 3D prior and maintain a persistent geometric state. The corrected geometry is fused with visual latents for prediction and planning, while an auxiliary objective encourages information captured by the SSM state to persist through the predictive architecture.
- We introduce **GOI-NAV**, a 3D navigation benchmark for evaluating planning success and physical path efficiency (SoftSPL) under **geometry-observation incongruence**, including displaced and unseen obstacles, deformable objects, and visual corruption.
- We demonstrate that **GeoMamba-WM substantially improves planning performance over DINO-WM** across both seen and zero-shot environment discrepancies, showing that even imperfect static 3D priors can provide useful grounding for latent world-model planning without requiring online dense 3D reconstruction.

2 Related Work

Pixel Reconstruction World Models World models predict the next state of an environment given prior observations and actions (Ha & Schmidhuber, 2018). Traditionally, these models use variational autoencoders (Kingma & Welling, 2014) or generative diffusion (Ho et al., 2020; Ding et al., 2024) to reconstruct future frames in pixel space. 3D-grounded models like PERSIST (Garcin et al., 2026) and 3D-Belief (Yin et al., 2026) additionally maintain dense 3D representations of the environment, requiring compute and latency that are prohibitive for real-time control.

World Models for Navigation Architectures like Dreamer (Hafner et al., 2025) use recurrent state space models (RSSMs) to maintain continuous hidden states throughout observations, and additionally train actor-critic policies for 3D navigation, requiring expensive online environment interactions for data collection and explicit reward engineering during training. Recently, models like NeDreamer (Bredis et al., 2026) employ decoder-free inference by removing the pixel decoder and planning entirely in the latent space, enabling real-time control with world models.

Latent World Models Latent world models push this further by predicting targets entirely within a compact latent space, distilling the information necessary to capture physical and dynamic environment properties while reducing latency. Joint embedding architectures like PLDM and LeWM jointly train the visual encoder alongside the predictor, while DINO-WM leverages a frozen pretrained DINOv2 encoder. During inference, these models employ zero-shot trajectory optimization techniques such as the Cross-Entropy Method (CEM) (Kroese et al., 2013) for planning, thus being trainable without online RL interactions or custom reward engineering. However, standard latent world models lack the explicit spatial memory required to navigate 3D scenarios reliably.

Explicit Mapping for Navigation Visual simultaneous localization and mapping (SLAM) maps unseen environments from only RGB images. Methodologies range from classical feature-based geometric point tracking (Campos et al., 2021) to deep optical flow representations (Teed & Deng, 2021) and transformer-based architectures that predict 3D point maps directly (Murai et al., 2025). Recent neural SLAM paradigms have evolved from implicit Neural Radiance Fields (NeRFs) (Zhu et al., 2022) and explicit 3D Gaussian Splatting (Matsuki et al., 2024) for real-time, photorealistic spatial representations. These methods use priors from previously mapped environments and update them with observations of dynamic and deformable obstacles. Navigation can then be performed live using graph-based algorithms such as A*, as is also done by world models like 3D-Belief that retain explicit maps. However, building and maintaining maps make planning computationally expensive.

Existing Benchmarks Standard visual control benchmarks for latent world models such as OpenAI Gymnasium (Brockman et al., 2016) and DeepMind Control Suite (Tassa et al., 2018) evaluate

agents using camera-static overviews rather than agent-centric observations. Popular egocentric environments like Habitat-Sim (Savva et al., 2019), however, are unable to model deformable obstacles for complex agent-environment interactions. 3D navigation in the real-world is necessarily dynamic, requiring models to reconcile cross-modal discrepancies between a stale, pre-mapped 3D prior and live observations. This leaves a critical gap in evaluating real-world robustness to displaced obstacles, deformable surfaces, and sensor noise.

3 METHODS

3.1 LATENT WORLD MODELS

Latent world models decouple environment dynamics from pixel-level reconstruction by operating directly in a latent space. Adapting the DINO-WM architecture, our model employs a frozen pre-trained DINOv2 encoder, then mean-pools its token outputs to obtain a compact latent $z_{o,t} = \text{MeanPool}(E_{\text{DINO}}(o_t)) \in \mathbb{R}^{384}$ from the high-dimensional observation o_t. The predictor P_θ is an action-conditioned causal transformer, modeling forward dynamics to estimate the subsequent latent state. The conditioning action a_t is injected into the predictor using adaptive layer normalization (AdaLN). The predictor is trained to minimize the L2 distance between the predicted and target latent encoded from the true next observation:

$$\hat{z}_{o,t+1} = P_\theta(z_{o,t}, a_t), \qquad \mathcal{L}_{\text{Pred}} = ||\hat{z}_{o,t+1} - \text{sg}(\text{MeanPool}(E_{\text{DINO}}(o_{t+1})))||_2^2$$

where $\text{sg}(\cdot)$ denotes a stop-gradient that prevents encoder updates from the optimization target.

3.2 PRETRAINED POINT CLOUD GEOMETRY ENCODER

The system assumes that a pre-existing 3D geometric map and agent position will be available for sampling during training and inference. To inject geometric information from this prior into the latent space, we pretrain and freeze a scene-specific geometry encoder E_G. It uses a multi-layer transformer architecture, learning 16 input query tokens for a $Z_g \in \mathbb{R}^{16\times384}$ output latent from 1024 sampled input points in a 1 meter radius. The use of a custom geometry encoder enables the ingestion of semantic point labels such as walkability and deformability. E_G is pre-trained via a point cloud autoencoder using Chamfer loss, along with binary cross entropy (BCE) losses for the walkability and deformability attributes. Detailed architecture and pre-training objectives are deferred to Appendix A.2.2.

3.3 SSM-BASED GEOMETRY CORRECTOR

To capture inconsistencies in the environment, we introduce a Mamba-based SSM geometry corrector that maintains a hidden state H_t and is updated using geometry, observation, and action inputs. The hidden state $H_0 = \phi_{\text{init}}(Z_{g,0})$ is initialized using the geometry latent at the initial timestep. We then construct input vectors X_t and I_t for the SSM:

$$X_t = [\text{LN}(Z_{g,t}), \text{LN}(H_{t-1}), E_{o,t}, E_{a,t-1}], \quad I_t = [\text{LN}(Z_{g,t}), E_{a,t-1}, E_{o,t}]$$
$$E_{o,t} = \tanh(\alpha)(\text{CrossAttn}(Z_{g,t}, Z_{o,t}) - Z_{g,t}), \qquad E_{a,t-1} = \phi_a(a_{t-1})$$

where $E_{o,t}$ is a residual embedding produced by an observation-conditioning block that ingests the observation and geometry latents via cross-attention. The gate $\tanh(\alpha)$, with learned scalar α, scales the residual by a factor constrained by $[-1, 1]$, and $E_{a,t-1}$ is the action embedding produced by an MLP encoding the previous action input.

We correct the geometry using three learned blocks A, B, and C. Block A predicts a token-wise retention factor, controlling how much of the hidden geometry state is retained. Block B predicts a gated candidate update. These are used to update the hidden state:

$$A_t = f_A(X_t), \qquad B_t = f_B(I_t, X_t), \qquad H_t = A_t \odot H_{t-1} + (1 - A_t) \odot B_t$$

where $\odot$ denotes element-wise multiplication. Block C reads the updated state and provides a bounded residual correction to produce the corrected geometry latent. Further specific implementation details are available in Appendix A.2.1.

$$C_t = f_C(H_t, X_t), \qquad \hat{Z}_{g,t} = Z_{g,t} + \gamma C_t$$

3.4 Cross-Attention Latent Fusion

The corrected geometry latent is fused with the observation latent in a cross-attention block, producing a fused latent that acts as the latent space of the world model.

$$Z_{f,t} = \text{Fusion}(Z_{o,t}, \hat{Z}_{g,t}) \in \mathbb{R}^{256\times384}, \qquad z_{f,t} = \text{MeanPool}(Z_{f,t}) \in \mathbb{R}^{384}$$

The predictor is an action-conditioned causal transformer, predicting fused latents instead of just observation latents. The target fused latent is therefore obtained by maintaining an exponential moving average (EMA) of the fusion block. The DINOv2 and geometry encoders are frozen, while stop-gradient prevents optimization through the target branch.

$$z^{\text{target}}_{f,t+1} = \text{sg}[\text{MeanPool}(\text{Fusion}_{\text{EMA}}(Z_{o,t+1}, Z_{g,t+1}))]$$

3.5 Hidden State Auxiliary Heads

To further enforce the inconsistency signal between geometry and observation within the fused latent space, we implement auxiliary heads to predict the SSM's hidden state from the encoded and predicted latents. Propagation of such auxiliary losses through the encoder and predictor in jointly-trained JEPA-style world models has been shown to improve structuring of their learned latent spaces to better capture quantities useful for their specific prediction task. Similarly to its implementation in Zhu et al. (2026), we use a 2-layer MLP to calculate auxiliary losses for each latent state:

$$\mathcal{L}^{\text{enc}}_{\text{Aux}} = ||\bar{h}^{\text{EMA}}_t - A_\psi(z_{f,t})||^2_2, \qquad \mathcal{L}^{\text{pred}}_{\text{Aux}} = ||\bar{h}^{\text{EMA}}_{t+1} - A_\psi(\hat{z}_{f,t+1})||^2_2$$

Where $\bar{h}^{\text{EMA}}_t$ and $\bar{h}^{\text{EMA}}_{t+1}$ are normalized, mean-pooled EMA hidden states. The auxiliary head itself is only trained on $\mathcal{L}^{\text{enc}}_{\text{Aux}}$, while $\mathcal{L}^{\text{pred}}_{\text{Aux}}$ is propagated into the world model objective using hyperparameter λ_A.

$$\mathcal{L}_{\text{WM}} = ||\hat{z}_{f,t+1} - z^{\text{target}}_{f,t+1}||^2_2 + \lambda_A \mathcal{L}^{\text{pred}}_{\text{Aux}}$$

3.6 Trajectory Planning

To evaluate model performance, the Cross-Entropy Method (CEM) (Kroese et al., 2013) is used for trajectory optimization within the learned fused latent space, identically to the setup used by DINO-WM. The predictor rolls out a series of latent states using an initial observation o_1 and a randomly sampled candidate action sequence $a_{1:H}$ up to the planning horizon H. A fixed number of elites are chosen based on the CEM cost between the predicted and goal latents as shown below, and the action sampling statistics are iteratively updated to improve the planning objective. The planner forms the mean action sequence of the final elites, executes only its first action, and replans after receiving the next observation.

$$C(\hat{z}_H) = ||\hat{z}_H - z_{\text{goal}}||^2_2, \quad a^*_{1:H} = \arg\min_{a_{1:H}} C(\hat{z}_H)$$

4 Benchmark

To evaluate our model on dynamic, pre-mapped 3D navigation, we introduce the Geometry-Observation Inconsistency Navigation (GOI-NAV) Benchmark, an egocentric 3D navigation benchmark built on the interactive, multimodal ThreeDWorld (TDW) simulator (Gan et al., 2021).

To rigorously test the model's capacity for cross-modal inconsistency, GOI-NAV is built upon a single base environment of a bounded square room with a central barrier. By fixing the default topology to a simple layout, we ensure that planning evaluations strictly measure the agent's ability to navigate with geometry-observation incongruence rather than standard pathfinding complexity.

A static point cloud of the base environment at $t = 0$ is passed to the model, labelled with walkability and deformability where appropriate. The agent operates using depth-less RGB vision from an egocentric camera, executing continuous actions a_t. The evaluation task is simple point-to-point navigation from a starting point to a visually identifiable goal marker, with success defined as reaching a fixed radius around the goal within a maximum episode step budget. Inconsistencies are introduced through three controlled axes of variations that perturb the default aligned geometry and observation:

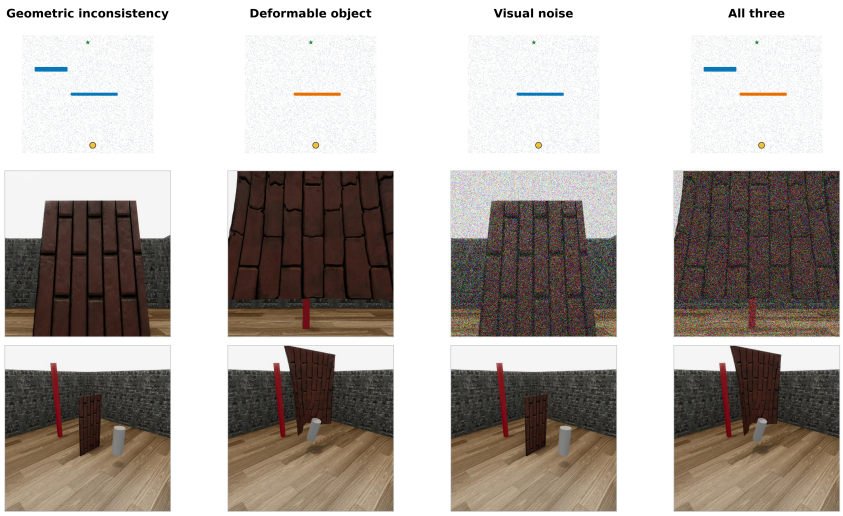

Figure 2: Sample 3D priors, egocentric observations, and third person views from the GOI-NAV benchmark with various inconsistencies.

- **Geometric Inconsistency** We force spatial discrepancy between the 3D prior and environment in two ways: **(1)** altering the environment by shifting the barrier or introducing new static or dynamic obstacles, and **(2)** shifting or adding geometry to the prior that is not present in the live RGB observation. These variations evaluate the model's ability to dynamically overwrite outdated geometric priors to avoid collisions and exploit new paths.
- **Deformable Objects** Another axis of variation is the introduction of deformable objects, whereby the environmental geometry becomes inconsistent with the prior upon interaction with the agent. The use of the semantic label available in the point cloud can measure the model's ability to use this information to interact appropriately, while omitting it can measure the harder task of dynamically learning deformations at planning-time.
- **Visual Noise** Inconsistencies can also stem from visual noise that forces the model to rely on the geometric prior for navigation when the observation is unreliable. This measures the model's ability to dynamically select between information sources to plan effectively.

To quantify planning robustness, we create a compositional evaluation set consisting of one, two, or all three of these variational axes, providing a holistic measure of the models' planning performance with geometry-observation inconsistencies. Additionally, we report success rates on a baseline of fully aligned scenes. For all evaluations, the benchmark reports two metrics:

- **Planning Success Rate** As described in section 3.6, the CEM planning protocol is used to optimize trajectories using each model with the exact same held-out evaluation data set. Success is defined as reaching within a 0.35m radius of the goal within 60 action blocks.
- **SoftSPL** Success rate alone fails to penalize poor path efficiency, as long, winding paths are undesirable in a real-world 3D navigation context. Success weighted by Path Length (SPL) is a metric that measures the ratio of an oracle shortest path to the model's path from the start point to the goal and is used to assess planning efficiency. Although SPL assigns unsuccessful paths a score of 0, SoftSPL (Datta et al., 2021) is a variation that instead weighs paths by their final geodesic distance from the goal, therefore partially rewarding unsuccessful rollouts that follow near-optimal trajectories. SoftSPL can indicate whether the model learns to exploit deformable surfaces to reduce path length.

5 Experimental Setup

5.1 Curriculum Training Procedure

Training the model simultaneously on aligned and inconsistent geometry-observation pairs can result in latent space collapse. Therefore, a two-stage curriculum learning setup is used to stabilize the fusion and predictor blocks. First, aligned-only environment data is used to pre-train the model to learn basic spatial kinematics and navigation. Second, we finetune the model using a curriculum that introduces a low ratio of variational data consisting of geometric inconsistencies, deformation, and visual noise. To prevent catastrophic forgetting, for aligned training samples we employ a latent teacher loss using the frozen aligned-only model, where z_{fused} is the encoded latent, $\hat{z}_{\text{fused}}$ is the predicted latent, and h is the hidden state of the student S and teacher T. For baselines and ablations without the hidden state, that term is excluded from the loss.

$$\mathcal{L}_{\text{teacher}} = \lambda_e ||z^S_{\text{fused}} - z^T_{\text{fused}}||^2_2 + \lambda_p ||\hat{z}^S_{\text{fused}} - \hat{z}^T_{\text{fused}}||^2_2 + \lambda_h ||h^S - h^T||^2_2$$

5.2 Baselines

We evaluate our method against latent world model baselines, controlled by using the same training data and curriculum, action space, CEM planner, and terminal latent-distance objective as GeoMamba-WM. These baselines are LeWM, PLDM, DINO-WM, and adaptations of each with frozen geometry encoders. We also evaluate navigation systems using their native control mechanisms. We implement NeDreamer and PERSIST-S with actor-critic planners, requiring privileged distance-based reward signals and online RL interactions, and 3D-Belief with its explicit map-based planner. Results from this latter group measure end-to-end navigation performance and establish reference comparisons for this environment, but do not isolate the contributions of the learned representation or dynamics model of GeoMamba-WM. Appendices A.3 and A.4 provide full implementation and adaptation details, along with additional comparisons.

6 Results

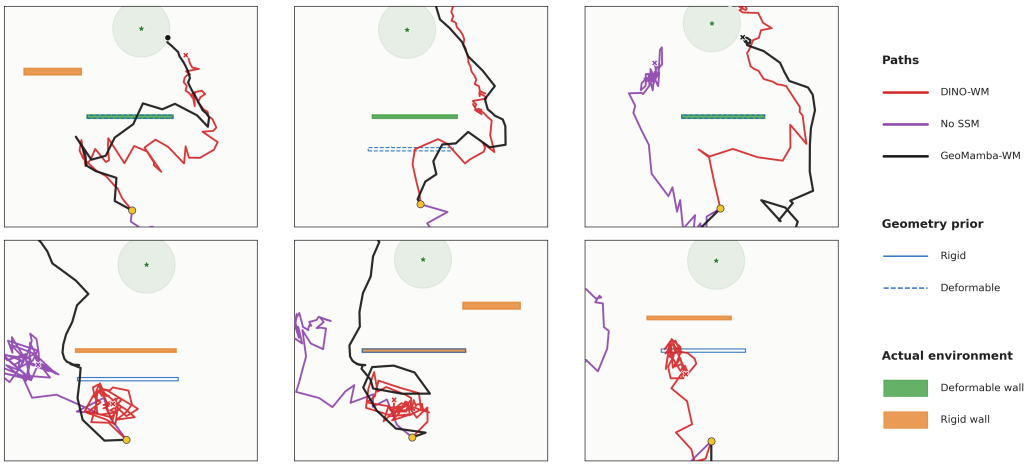

Figure 3: Sample paths taken by DINO-WM, our no SSM ablation, and GeoMamba-WM. Each diagram demonstrates cases where GeoMamba-WM was able to reason about inconsistencies in the environment to cross through deformable barriers, ignore invalid geometry priors, or navigate around surprise obstacles to get closer to the goal than the other models.

To evaluate the performance of our method in planning with geometry-observation incongruence, we report planning success rates and SoftSPL on our GOI-NAV benchmark's aligned and aggregate inconsistent evaluation sets, along with overlapping marginal metrics for each axis of inconsistency. Full implementation details, extended variance metrics, and additional baseline comparisons are available in Appendices A.3 and A.4.

6.1 Comparison Against Latent World Model Baselines

Table 1: Navigation performance on the GOI-NAV benchmark. Each entry reports average success rate (SR) and SoftSPL in percentages across 3 evaluation seeds.

	Aligned		Inconsistent		Geometry Marginal		Deformation Marginal		Noise Marginal	
Method	SR	SoftSPL	SR	SoftSPL	SR	SoftSPL	SR	SoftSPL	SR	SoftSPL
PLDM	0.0	4.0	0.2	0.7	0.0	0.7	0.0	0.0	0.2	0.6
PLDM + Geometry	0.0	6.0	3.9	3.8	4.4	3.9	2.4	1.5	2.3	3.2
LeWM	0.0	0.7	0.0	0.2	0.0	0.1	0.0	0.0	0.0	0.1
LeWM + Geometry	0.0	1.4	2.1	2.5	1.9	2.3	4.2	2.8	2.6	2.7
DINO-WM	4.4	9.0	2.5	6.1	2.3	6.0	2.4	6.3	2.8	5.3
DINO-WM + Geometry	33.3	29.1	27.0	17.3	25.4	17.2	28.1	14.5	23.4	16.1
GeoMamba-WM (Aligned Data Only)	**55.6**	**36.8**	27.3	14.7	25.0	14.2	32.3	13.4	25.0	13.5
GeoMamba-WM (Variational Finetune)	37.8	33.0	**37.1**	**18.8**	**33.8**	**17.9**	**43.1**	**17.3**	**36.3**	**18.4**

Results demonstrate the extreme difficulty of continuous 3D control using only offline, reward-free dynamics. Standard latent world model baselines lacking geometric priors are unable to navigate to the goal almost entirely. Augmenting LeWM and PLDM with geometry improves their planning success marginally, while DINO-WM improves greatly with the frozen geometry encoder, indicating that geometric priors are critical for success in 3D navigation tasks. GeoMamba-WM, when trained on fully aligned data, achieves a much higher 55.6% success rate in aligned scenarios. Performance degrades by half on the inconsistent evaluation set, but its success still demonstrates the ability of our model to adapt to unseen dynamic environments. Fine-tuning GeoMamba-WM on variational data greatly improves planning with cross-modal inconsistency, increasing inconsistent success rates from 27.3% to 37.1%. We observe a trade-off of aligned evaluation success rate dropping from 55.6% to 37.8%, although the SoftSPL degrades only slightly from 36.8% to 33.0%, indicating that forgetting is not catastrophic due to the teacher loss described in section 5.1.

6.2 Geometry and SSM Ablations

Table 2: Performance comparisons between various geometry encoders and SSM implementations, reported as an average of 3 CEM seeds. Standard deviations are available in Appendix A.11.

	Aligned		Inconsistent		Geometry Marginal		Deformation Marginal		Noise Marginal	
Method	SR	SoftSPL	SR	SoftSPL	SR	SoftSPL	SR	SoftSPL	SR	SoftSPL
No SSM	33.3%	29.1%	27.0%	17.3%	25.4%	17.2%	28.1%	14.5%	23.4%	16.1%
SSM, No Aux Head	23.3%	29.0%	33.5%	**19.2%**	29.4%	**18.4%**	36.1%	15.4%	31.5%	**18.6%**
Observation-Only SSM, No Aux Head	6.7%	19.5%	5.7%	10.3%	5.2%	10.1%	2.4%	4.5%	6.5%	9.8%
DINO-WM	*4.4%*	*9.0%*	*2.5%*	*6.1%*	*2.3%*	*6.0%*	*2.4%*	*6.3%*	*2.8%*	*5.3%*
Ours (SSM, Aux Head)	**37.8%**	**33.0%**	**37.1%**	18.8%	**33.8%**	17.9%	**43.1%**	**17.3%**	**36.3%**	18.4%

To isolate the structural contributions of our architecture, we ablate the use of the geometry encoder and SSM block, all without any auxiliary heads and fine-tuned with variational data. Removing the SSM completely yields a fusion model that cannot correct for dynamic geometry. Although it improves over the DINO-WM baseline, it cannot resolve inconsistencies as effectively as the SSM, achieving a lower success rate with inconsistencies. Conversely, as shown by the observation-only SSM, relying only on recurrent memory without geometry input results in severe performance collapse, indicating that geometric information is mandatory for effective 3D navigation.

Table 3: Percentage of all deformable scenarios where the agent crossed the barrier.

Method	Crossing Rate
DINO-WM	0.00%
No SSM	4.17%
Base SSM (No Aux Head)	10.42%
GeoMamba-WM (Ours)	**17.71%**

Table 4: Pearson correlation (r) between latent representations and environment wall position.

Target	Fused (r)	Hidden (r)
Physical wall X	0.891	0.769
Physical wall Z	0.956	0.843
X mismatch	0.096	0.079
Z mismatch	0.265	0.550

Implementing the SSM block with a geometry encoder significantly boosts inconsistent success rates and path efficiency by accounting for changing geometry during navigation. Table 3 shows how models with an SSM cross deformable walls at higher rates. Visualized paths in Figure 3 reveal

that the SSM captures deformation through visual observations and corrects the geometric prior, while the No-SSM model treats the wall as rigid and navigates inefficiently around it. The hidden state auxiliary target raises wall crossing rates a further 7.29% over the base SSM by reinforcing this corrected geometry state within the fused latent space. We train linear probes on frozen fused latents and hidden states to predict the wall position, along with mismatches $X_{\text{mismatch}} = X_{\text{physical}} - X_{\text{map}}$ for both X and Z. Results in table 4 find that the hidden state has a strong correlation with the mismatch in the z-coordinate of the wall, representing a forward-backward shift in its location, which is propagated into the fused latent and used by the planner in scenarios with geometric inconsistency.

6.3 Comparison Against Other End-to-End Planning Models

To contextualize our results, we evaluate GeoMamba-WM against systems utilizing their native, task-specific controllers. First, we observe the constraints of existing paradigms: NeDreamer achieves an 88.9% aligned success rate through online RL, but its low SoftSPL, even below that of GeoMamba-WM with CEM, reveals highly inefficient, winding path execution. 3D-Belief achieves efficient paths by maintaining an explicit, online 3D scene and diffusion-based view imagination, but incurs a 22.6s per-action latency that is prohibitive for real-time navigation. These demonstrate that GeoMamba-WM, using its standard CEM planner, already achieves competitive real-time physical reasoning strictly in the latent space without relying on explicit mapping or online RL.

To provide a fairly matched comparison, we also trained Soft Actor-Critic (SAC) planners on frozen backbones for both PERSIST-S and GeoMamba-WM, matching the online collection protocol and distance-based rewards used by NeDreamer. GeoMamba-WM with SAC achieves a 100% aligned success rate with higher SoftSPL than PERSIST, and remains competitive in inconsistent scenarios, accomplishing this with $17\times$ less inference latency and over $164\times$ less compute. This confirms that GeoMamba-WM captures rich, actionable spatial representations that scale to state-of-the-art (SOTA) performance when provided with task-specific reinforcement learning.

Table 5: End-to-end navigation performance and system cost on the GOI-NAV benchmark. Performance values are three-seed means in percentages. All costs are measured on an NVIDIA A6000.

		Aligned		Inconsistent		Training Cost			Inference Cost	
Method	Controller	SR	SoftSPL	SR	SoftSPL	Collection	Transitions	Train Time	Latency	GFLOPs
NeDreamer	Actor-Critic	88.9	25.9	68.8	22.2	Online	25k	6.6 h	3.5 ms	0.16
3D-Belief	Belief + A*	86.7	51.9	40.8	26.8	Offline	4.98k	1.0 h	22.6 s	127,567.9
PERSIST-S + SAC	Actor-Critic	**100**	<u>82.0</u>	**94.4**	**56.3**	Online	25k	15.4 h	292.2 ms	4001.1
GeoMamba-WM	CEM	37.8	33.0	37.1	18.8	Offline	11.05k	0.2 h	200.1 ms	299.4
GeoMamba-WM + SAC	Actor-Critic	**100**	**83.4**	<u>86.4</u>	<u>47.8</u>	Online	25k	5.7 h	17.0 ms	24.4

7 Conclusion

We introduced GeoMamba-WM, a geometry-grounded latent world model capable of navigating in dynamic 3D environments with static geometric priors. By integrating a Mamba-based State Space Model (SSM), our architecture establishes a persistent hidden state that corrects stale pre-mapped geometry using live RGB observations completely in the latent space. To evaluate this, we introduced GOI-NAV, a 3D navigation benchmark focused on navigation with geometry-observation incongruence. Evaluations demonstrate that GeoMamba-WM avoids the catastrophic representation collapse seen in standard latent world model baselines, successfully handling geometric inconsistencies, deformations, and visual noise by using its SSM to correct geometric priors. We also compared our model with model-based RL and 3D-mapping baselines. While our reward-free architecture establishes a highly viable foundation for offline planning, we further demonstrated that when equipped with a task-specific actor-critic controller, GeoMamba-WM performs competitively with state-of-the-art generative world models using a fraction of the compute and latency. Our results demonstrate that latent world models are viable for planning in dynamic 3D environments using static geometry priors and RGB input, and that cross-modal inconsistencies can be resolved entirely in the latent space without computationally-heavy online reconstruction or mapping objectives.

Limitations and Future Work While GeoMamba-WM establishes a highly viable architecture for 3D navigation, GOI-NAV only evaluates with limited DoF, environments, and embodiments. Future work will explore expanding the GOI-NAV benchmark to support 6-DoF embodiments across multiple topologies and evaluations of GeoMamba-WM on more dynamic, real-world environments.

AI USE STATEMENT

Generative AI tools were used for assistance in writing and modifying code for environment generation, implementation of parts of the model, along with training and inference infrastructure. These tools were also used to aid with polishing the writing of this paper. The research ideas, methodology, model architectures, experimental design, and analysis of the results were conducted wholly by the authors. All code and text written by AI was thoroughly reviewed by the authors to prevent errors and inaccuracies. The authors take full responsibility for this work.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the model architecture, the benchmark and its design, and implementation details such as training and inference seeds and parameters, evaluation metrics and design, and ablation implementations within both the main text and appendix. The source code and custom benchmark, along with any model checkpoints will be released upon publication.

REFERENCES

George Bredis, Nikita Balagansky, Daniil Gavrilov, and Ruslan Rakhimov. Next embedding prediction makes world models stronger. In *ICLR 2026 the 2nd Workshop on World Models: Understanding, Modelling and Scaling*, 2026. URL https://openreview.net/forum?id=SkAgjqPmhY.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. URL https://arxiv.org/abs/1606.01540.

Carlos Campos, Richard Elvira, Juan J. Gomez Rodriguez, Jose M. M. Montiel, and Juan D. Tardos. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, December 2021. ISSN 1941-0468. doi: 10.1109/tro.2021.3075644. URL http://dx.doi.org/10.1109/TRO.2021.3075644.

Samyak Datta, Oleksandr Maksymets, Judy Hoffman, Stefan Lee, Dhruv Batra, and Devi Parikh. Integrating egocentric localization for more realistic point-goal navigation agents. In Jens Kober, Fabio Ramos, and Claire Tomlin (eds.), *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 313–328. PMLR, 16–18 Nov 2021. URL https://proceedings.mlr.press/v155/datta21a.html.

Zihan Ding, Amy Zhang, Yuandong Tian, and Qinqing Zheng. Diffusion world model: Future modeling beyond step-by-step rollout for offline reinforcement learning, 2024. URL https://arxiv.org/abs/2402.03570.

Chuang Gan, Jeremy Schwartz, Seth Alter, Damian Mrowca, Martin Schrimpf, James Traer, Julian De Freitas, Jonas Kubilius, Abhishek Bhandwaldar, Nick Haber, Megumi Sano, Kuno Kim, Elias Wang, Michael Lingelbach, Aidan Curtis, Kevin Feigelis, Daniel Bear, Dan Gutfreund, David Cox, Antonio Torralba, James J DiCarlo, Josh Tenenbaum, Josh McDermott, and Dan Yamins. Threedworld: A platform for interactive multi-modal physical simulation. In J. Vanschoren and S. Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/735b90b4568125ed6c3f678819b6e058-Paper-round1.pdf.

Samuel Garcin, Thomas Walker, Steven McDonagh, Tim Pearce, Hakan Bilen, Tianyu He, Kaixin Wang, and Jiang Bian. Beyond pixel histories: World models with persistent 3d state. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=yyitz0hJqK.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems 31*, pp. 2451–2463. Curran Associates, Inc., 2018. URL https://papers.nips.cc/paper/7512-recurrent-world-models-facilitate-policy-evolution. https://worldmodels.github.io.

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Dirk P. Kroese, Reuven Y. Rubinstein, Izack Cohen, Sergey Porotsky, and Thomas Taimre. *Cross-Entropy Method*, pp. 326–333. Springer US, Boston, MA, 2013. ISBN 978-1-4419-1153-7. doi: 10.1007/978-1-4419-1153-7_131. URL https://doi.org/10.1007/978-1-4419-1153-7_131.

Lucas Maes, Quentin Le Lidec, Luiz Facury, Nassim Massaudi, Ayush Chaurasia, Francesco Capuano, Richard Gao, Taj Gillin, Dan Haramati, Damien Scieur, Yann LeCun, and Randall Balestriero. stable-worldmodel: A platform for reproducible world modeling research and evaluation, 2026a. URL https://arxiv.org/abs/2605.21800.

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

Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. Gaussian splatting slam. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18039–18048, 2024. doi: 10.1109/CVPR52733.2024.01708.

Riku Murai, Eric Dexheimer, and Andrew J. Davison. Mast3r-slam: Real-time dense slam with 3d reconstruction priors. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16695–16705, 2025. doi: 10.1109/CVPR52734.2025.01556.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seung Eun Yi, Michael Ramamonjisoa, Francisco Massa, Daniel HAZIZA, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Herve Jegou, Patrick Labatut, and Piotr Bojanowski. DINOv3. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=2NlGyqNjns. Featured Certification.

Uladzislau Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 43905–43941. Curran Associates, Inc., 2025. doi: 10.52202/085713-1465. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/3e7cf447f21cd11c846463affefce665-Paper-Conference.pdf.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. Deepmind control suite, 2018. URL https://arxiv.org/abs/1801.00690.

Zachary Teed and Jia Deng. DROID-SLAM: Deep visual SLAM for monocular, stereo, and RGB-d cameras. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=ZBfUo_dr4H.

Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin. *Understanding and improving layer normalization*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Yifan Yin, Zehao Wen, Suyu Ye, Jieneng Chen, Zehan Zheng, Nanru Dai, Haojun Shi, Aydan Huang, Zheyuan Zhang, Alan Yuille, Jianwen Xie, Ayush Tewari, and Tianmin Shu. 3d-belief: Embodied belief inference via generative 3d world modeling, 2026. URL https://arxiv.org/abs/2605.11367.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: world models on pre-trained visual features enable zero-shot planning. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org, 2025.

Penghao Zhu, Salvatore Penachio, Kaustav Mukherjee, and Aneesh Jonelagadda. Spectral-target physical latent structuring for jepa-style world models, 2026. URL https://arxiv.org/abs/2609.04264.

Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12786–12796, June 2022.

A Appendix

A.1 GOI-NAV Benchmark

Geometry-Observation Inconsistency Navigation (GOI-NAV) evaluates navigation performance of models in pre-mapped 3D environments where visual observations conflict with geometry priors. The benchmark varies three axes of incongruence: geometric inconsistency, physical deformability, and visual corruption. Evaluations are performed without inconsistencies, with each axis on its own, and with combinations of these perturbations simultaneously.

A.1.1 Environment Setup

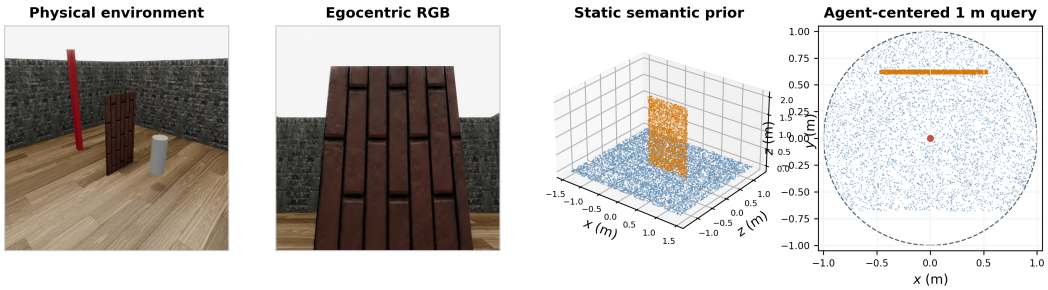

Figure 4: Overview of base benchmark environment, with examples of a rendered visualization of the environment, an egocentric RGB snapshot, and the static semantic point cloud prior and its corresponding query.

Simulator We implement GOI-NAV in ThreeDWorld (TDW), which provides image-based rendering, rigid-body dynamics, and an Obi-based cloth simulator, enabling the environment to contain both rigid and deformable barriers, as well as movable objects. All rollouts are simulated at 60 frames per second.

Agent Embodiment A single agent embodiment is used for this benchmark. It is a cylindrical rigid body with radius $0.16\,\mathrm{m}$, diameter $0.32\,\mathrm{m}$, height $0.8\,\mathrm{m}$, and mass $10\,\mathrm{kg}$. Its dynamic and static friction coefficients are both 0.4, with zero bounciness. Vertical translation is constrained to enforce planar navigation.

Action Space At each decision step, the agent selects a world-frame planar displacement

$$a_t = (\Delta x_t, \Delta z_t) \in \mathbb{R}^2, \qquad \|a_t\|_2 \le 0.40\,\mathrm{m}.$$

Each displacement is converted into a target position and executed through a rigid-body controller for 24 physics frames, corresponding to $0.4\,\mathrm{s}$ of simulated time, with a maximum drive speed of $2.0\,\mathrm{m/s}$ and drive force of 0.10.

Egocentric RGB Camera The RGB camera captures 256×256 observations with a $75°$ field of view. It is mounted at the relative position $(0, 0.72, 0.05)\,\mathrm{m}$ and pitched downward by $12°$. The camera follows the agent's translation but does not rotate with the rigid body. Consequently, both the camera and the action coordinates remain aligned with the world yaw axis. Images are resized to 224×224 before being passed to the frozen DINOv2 encoder, or to the required sizes for other baseline models.

Static Geometric Prior Each layout is associated with astatic semantic point-cloud map. It is generated once from the initial scene and not updated throughout the rollout. It contains points $p_i = x_i, y_i, z_i, w_i, d_i$, where $w_i \in \{0, 1\}$ indicates walkability and $d_i \in \{0, 1\}$ indicates deformability. Floor points have $w_i = 1$, while barrier and obstacle points have $w_i = 0$. For deformable scenarios, points belonging to the mapped primary barrier have $d_i = 1$, while floor points and additional rigid obstacles retain $d_i = 0$.

At each timestep, the static map is queried with a $1\,\mathrm{m}$ radius around the agent, producing a spherical point cloud input. 1024 points are sampled, using deterministic voxel-based subsampling when more points are available, and padding when fewer are. The absolute XYZ coordinates are transformed into an agent-centered coordinate frame, $\tilde{p}_{i,t}^{xyz} = p_i^{xyz} - (x_t, 0, z_t)$, but not rotated by the agent orientation. Thus, similar to the camera, the input point cloud is agent-centered but world-yaw aligned.

Base Layout The environment is an $8\,\mathrm{m} \times 8\,\mathrm{m}$ bounded room containing a central barrier near $z = 0$. The barrier has height $2.0\,\mathrm{m}$, thickness $0.04\,\mathrm{m}$, and a procedurally sampled width $r_w \sim \mathcal{U}(1.0, 1.3)\,\mathrm{m}$. Its horizontal center is sampled as $x_w \sim \mathcal{U}(-0.15, 0.15)\,\mathrm{m}$. The start and goal positions are sampled on opposite sides of the barrier:

$$\begin{aligned} x_s, x_g &\sim \mathcal{U}(-0.20, 0.20)\,\mathrm{m}, \\ z_s &\sim \mathcal{U}(-1.15, -1.05)\,\mathrm{m}, \\ z_g &\sim \mathcal{U}(1.05, 1.15)\,\mathrm{m}. \end{aligned}$$

Layouts are rejected unless the direct start-goal segment intersects the barrier and collision-free routes remain available around its endpoints. The navigation planner uses an additional $0.07\,\mathrm{m}$ clearance beyond the agent radius.

Inconsistencies are introduced on top of this base layout.

A.1.2 Geometric Inconsistency

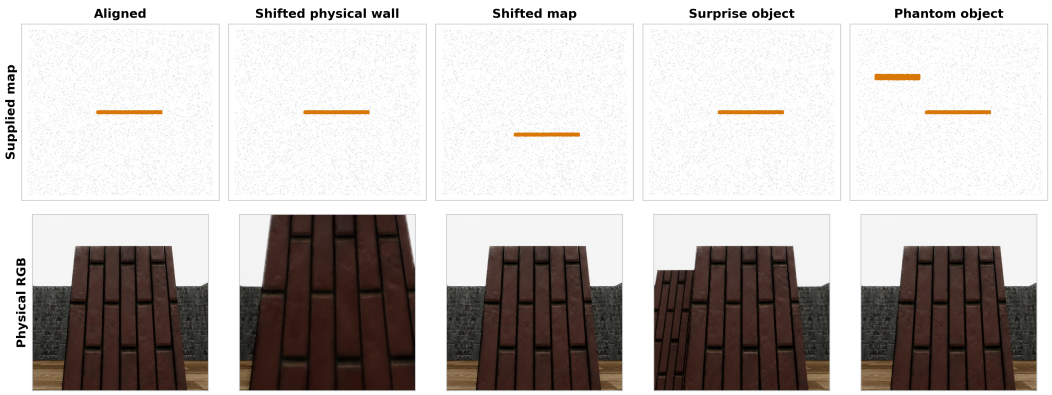

Figure 5: Overhead view and egocentric camera renders of geometric inconsistencies.

Geometric inconsistencies are introduced by misaligning the environment and 3D prior. There are five separate variations of geometric inconsistency.

Shuffled Physical Barrier In this condition, the physical barrier observed in RGB is translated in the x-z plane, while the supplied map retains the canonical barrier. For layout ℓ, the physical center is $(x_w^{\mathrm{phys}}, z_w^{\mathrm{phys}}) = (x_w^{\mathrm{map}}, 0) + \Delta_\ell$, where the displacement is bounded to $[-0.4\,\mathrm{m}, 0.4\,\mathrm{m}]$ for each of x and z.

Shuffled Map Barrier In this condition, the physical barrier observed in RGB remains canonical, while the barrier in the sampled 3D prior is shifted using the same logic as the physical shuffle barrier.

Surprise Static Physical Obstacle A rigid obstacle is inserted into the physical scene but omitted from the supplied map. The obstacle has width $0.7\,\mathrm{m}$, thickness $0.08\,\mathrm{m}$, and height $2.0\,\mathrm{m}$. It is placed at $z = 0.55\,\mathrm{m}$ and at

$$x_o = x_w + s_\ell(0.95\,\mathrm{m}), \qquad s_\ell \in \{-1, 0, 1\}.$$

The deterministic placement cycle assigns 40% of layouts to the left route, 20% to the center, and 40% to the right route.

Phantom Static Obstacle Identical to the surprise static physical obstacle, except the obstacle is included in the 3D prior and not the physical environment.

Surprise Dynamic Obstacle An unmapped rigid obstacle with width $1.10\,\mathrm{m}$ is added to the physical environment at $z = 0.75\,\mathrm{m}$. Its horizontal position follows $x_o(b) = 0.85\sin\left(\frac{2\pi b}{16}\right)\,\mathrm{m}$, where b is the action-block index. The object is absent from the static map, which contains no representation of its instantaneous position.

A.1.3 Deformable Objects

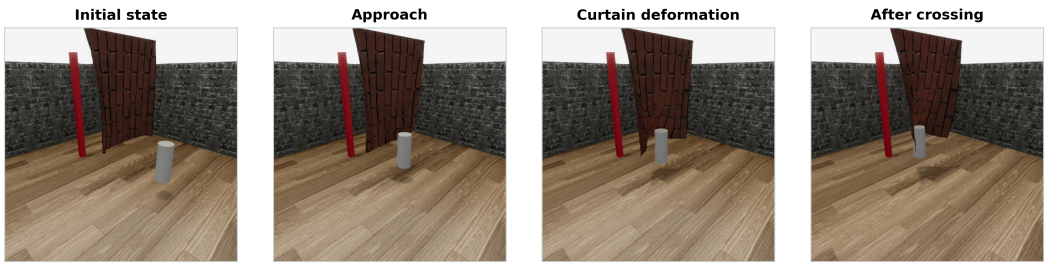

Figure 6: Rendered views of the agent passing through a deformable object.

When there are deformable objects within an environment, the geometric prior cannot capture the physical motion of the deformable object as the agent interacts with it, introducing inconsistency between the geometry and observation. For this environment, we introduce deformability by replacing the rigid central barrier with an Obi cloth sheet while preserving a visually matched brick texture, thus resembling a curtain.

The sheet uses the high-density cloth discretization and is tethered along its world-top edge. Its calibrated material parameters are:

$$\begin{aligned} \text{stretching scale} &= 1.0, \\ \text{stretch compliance} &= 0.0, \\ \text{maximum bending} &= 0.20, \\ \text{bend compliance} &= 0.02, \\ \text{mass per unit area} &= 0.30\,\mathrm{kg/m^2}, \\ \text{drag} &= 0.05, \\ \text{lift} &= 0.05. \end{aligned}$$

The agent-cloth static and dynamic friction coefficients are both 0.02. The rendered sheet uses a width compensation factor of 1.19 and a calibrated bottom-clearance parameter of $1.20\,\mathrm{m}$.

The supplied map remains static throughout an episode, with the mapped primary barrier labeled as deformable.

A.1.4 Visual Corruptions

Visual corruptions represent another axis of where RGB observations may differ from 3D geometric priors. Corruptions are applied to normalized RGB images in $[0, 1]$. Their parameters are sampled deterministically from the evaluation seed and held fixed at the sequence level, preventing significant frame-level fluctuation.

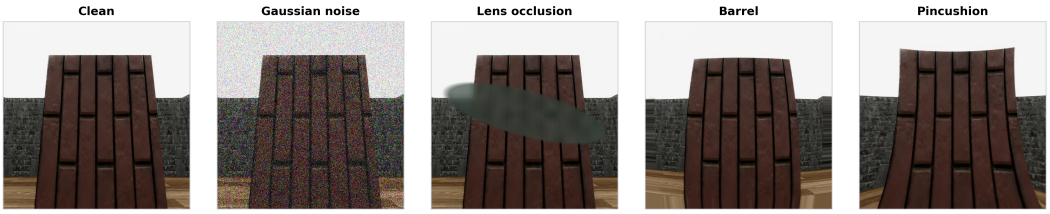

Figure 7: Egocentric rendered views of various visual corruptions added in the benchmark.

Gaussian Sensor Noise For each rollout, a noise level is sampled as $\sigma \sim \mathcal{U}(0.10, 0.40)$. Independent pixel and channel noise is then sampled at every timestep:

$$I'_t(x, y, c) = \text{clip}\left(I_t(x, y, c) + \epsilon_t(x, y, c), 0, 1\right), \qquad \epsilon_t(x, y, c) \sim \mathcal{N}(0, \sigma^2).$$

Thus, the magnitude σ is fixed over the rollout, while individual noise samples vary across pixels, channels, and frames.

Occlusion Occlusion is implemented in a lens smudge-like fashion, using a soft, textured elliptical artifact. Its aspect ratio is sampled log-uniformly from $[2, 5]$, its angle uniformly from $[-20°, 20°]$, and its opacity uniformly from $[0.90, 0.99]$. The horizontal center is sampled uniformly across the image; the vertical center follows a clipped Gaussian centered at the image midpoint with standard deviation 0.23 times the image height. The boundary uses a feather fraction sampled from $[0.08, 0.18]$.

Radial Lens Distortion Either barrel or pincushion distortion is applied to the RGB observation, with the two balanced across layouts. Its severity is sampled only once per layout, $|s| \sim \mathcal{U}(0.03, 0.12)$. For normalized image coordinates $(x, y) \in [-1, 1]^2$ and $r^2 = x^2 + y^2$, the inverse sampling grid is $(x', y') = (x, y)(1 - sr^2)$. Bilinear sampling and border padding are used, corresponding to approximately 4 to 15 pixels of horizontal displacement at the image boundary. Distortion parameters remain constant throughout a rollout.

A.1.5 Training Data Generation

Training data is pre-generated for this benchmark, intended for use in an offline fashion.

Behavior Policies The raw canonical rigid dataset contains five privileged data-collection policies per layout:

1. the shortest collision-free detour;
2. the alternate-side detour;
3. a perturbed shortest detour;
4. a perturbed alternate detour; and
5. a direct contact-probe trajectory.

The noisy policies perturb intermediate waypoints with zero-mean Gaussian noise having standard deviation $0.035\,\text{m}$, provided that the resulting route remains collision-free. The contact-probe controller moves directly toward the goal and intentionally attempts interaction with the central barrier. This combination of policies ensures that the world model is able to learn the physics of the environment effectively, instead of just memorizing successful paths.

These oracle rollouts are collected by first creating a two-dimensional occupancy grid from the privileged geometry state at a resolution of $0.05\,\text{m}$. Obstacles are inflated by the agent radius and an additional planning clearance. An A* planner computes collision-free paths from the sampled start position to the goal. This resulting grid path is compressed into world-frame keypoints. At each control step, the oracle policy selects the next waypoint and issues a bounded planar displacement, and a low-level force controller executes this displacement over a fixed number of simulator frames.

For environments with surprise physical obstacles that block one of the two routes, some paths collide with the obstacle but are still retained to ensure the model learns relevant contact physics. For deformable environments, a greater proportion of collision probes are used so the model can appropriately learn contact dynamics. Additionally, the push direction for collision is perturbed to have three approaches: direct, oblique right, and oblique left.

Physical Data Sources The benchmark contains eight physical source families formed by four geometric configurations: aligned, shuffled barrier, static obstacle, and dynamic obstacle. Each of these are conditioned with and without deformability, therefore forming eight total physical environments. The trajectories are reused for corresponding noised and geometry prior altered conditions, as the physical environment and thus optimal trajectories remain the same.

The original aligned rigid source contains 450 combined training and validation episodes, corresponding to five policies on 90 layouts. Each variation source contains 270 episodes over 90 layouts, with one detour, one noisy detour, and one contact-probe trajectory selected per layout.

Matched Aligned Pretraining Set The aligned dataset currently does not just contain the base environment and map, but three different aligned conditions:

1. a canonical rigid barrier with its matching canonical map;
2. a shuffled rigid barrier with a correspondingly shuffled map; and
3. a rigid static obstacle with a correspondingly augmented map.

All three of these combinations satisfy the aligned condition and prevent the model from learning a single pre-determined path. A total of 450 episodes are used for the training process, 400 as training and 50 as validation.

Layout Splits and Trajectory Reuse Offline world model training uses 80 layouts, while ten layouts are used for validation. 30 more held-out test layouts are used for evaluation. Whenever a physical trajectory is reused across inconsistent conditions, all derived versions retain the same split assignment, meaning all test layouts are unseen by the model during training.

A.1.6 Evaluation Composition

The goal of the evaluation setup is to ensure the model can pass with reasonable success rates in aligned situations, and then carry over that performance when inconsistencies are introduced. Therefore, the 30 layouts are used for an aligned benchmark, with 10 each for each of the aligned conditions described above.

The remaining evaluations are conducted on combinations of one, two, or three inconsistencies. This yields 47 total inconsistent combinations, and using four layouts for each yields 188 final evaluations.

Disjoint group	**Conditions**	**Jobs**
Aligned	1	30
Geometry-only	5	20
Deformation-only	1	4
Noise-only	3	12
Two-factor	23	92
Three-factor	15	60
Total		**218**

For the factor-wise columns in the main table, we compute overlapping marginals over the 188 inconsistent jobs to demonstrate success in scenarios with that variation.

Reported marginal	Jobs
Geometry active	160
Deformation active	96
Noise active	144
Multi-factor	152
Overall inconsistent	188

Evaluation Randomization To ensure fairness in reported results given the stochasticity of the Cross-Entropy Method (CEM) for planning, we report mean and standard deviation of results across three evaluation seeds: 42, 43, and 44. The same layout, conditions, corruption seed construction, start, goal, and seeds are used for every model evaluated.

For deterministic actor-critic policies, the actor itself does not sample actions during evaluation and thus cannot be randomized. For NeDreamer, the RSSM posterior state is sampled at every iteration, meaning the evaluation seed changes the inferred latent trajectory. For PERSIST and 3D-Belief, the evaluation seed affects noise sampling during diffusion. For GeoMamba-WM with an SAC controller, the main source of variation is from simulator-level execution variability from non-deterministic physics calculations in the TDW simulator, which also applies to all other evaluations.

A.1.7 Evaluation Metrics

Two evaluation metrics are used for this benchmark to ensure model performance is described holistically.

Success Rate Let $q_{i,t} \in \mathbb{R}^2$ be the physical agent position and g_i the goal for rollout i. A rollout succeeds if the physical simulator reports

$$\|q_{i,t} - g_i\|_2 \leq \tau, \qquad \tau = 0.35\,\mathrm{m},$$

at any point within at most 60 action blocks. Evaluation terminates immediately after the first successful block. Success Rate measures the percentage of rollouts where this is achieved.

SoftSPL To better capture model behavior, we also report SoftSPL, which is particularly useful in determining a model's ability to exploit deformable surfaces to reduce the taken path length. SoftSPL rewards partial geodesic progress while penalizing inefficient paths with respect to the oracle.

For rollout i, let d_i^0 be the physical-scene geodesic distance from the initial position to the goal, d_i^T the geodesic distance from the final position to the goal, and L_i the physical path length actually traversed. We define progress and efficiency as

$$P_i = \max\left(0, 1 - \frac{d_i^T}{d_i^0}\right), \qquad E_i = \frac{d_i^0}{\max(d_i^0, L_i)}.$$

The reported metric is

$$\text{SoftSPL} = \frac{1}{N}\sum_{i=1}^{N} P_i E_i.$$

Unlike standard SPL, SoftSPL is not multiplied by a binary success indicator. Consequently, an unsuccessful rollout receives credit when it makes genuine progress toward the goal, but that credit is reduced when it follows an inefficient path with respect to the geodesic oracle.

This geodesic oracle uses a simple occupancy grid with resolution $0.05\,\mathrm{m}$ that is constructed from the physical scene, with rigid barriers and static surprise obstacles marked as occupied, and an A* search is performed on this grid to determine the oracle path. Deformable objects remain unoccupied, rewarding models for exploiting deformable objects' traversability. Dynamic obstacles are also unmarked as the oracle does not perform space-time planning, meaning that the resultant geodesic is an optimistic lower-bound and not a true shortest achievable path in all scenarios. Meanwhile, L_i is measured from the agent's executed trajectory, and therefore includes collisions and detours induced by all obstacle types.

A.2 Implementation Details

A.2.1 SSM Geometry Corrector

The geometry corrector maintains a token-wise recurrent state $h_t \in \mathbb{R}^{K\times d}$, where $K = 16$ geometry tokens and $d = 384$. Its purpose is to update the static-map representation $Z_{g,t} \in \mathbb{R}^{K\times d}$ using the current visual observation $Z_{o,t} \in \mathbb{R}^{N\times d}$ and the previously executed action $a_{t-1} \in \mathbb{R}^2$. All operations below are applied independently to each geometry token, with learned parameters shared across tokens and time.

State Initialization At the first observation, the hidden state is initialized using the geometry representation.

$$h_0 = W_{\text{init}}\text{LN}_g(Z_{g,0}) + b_{\text{init}}$$

Geometry-Aligned Observation Input Because the visual and geometry latents contain different numbers of tokens, the observation is projected into the geometry-token structure.

$$Z_t^{o\to g} = \text{CrossAttentionBlock}(\text{query} = Z_{g,t}, \text{keys} = Z_{o,t}, \text{values} = Z_{o,t})$$

The attention block uses four heads, dropout 0.1, and an MLP expansion ratio of four. We isolate the update proposed by this residual cross-attention block:

$$E_{o,t} = \tanh(\alpha)\,(Z_t^{o\to g} - Z_{g,t})\,,$$

Where α is a learned scalar initialized to 0.1. The $\tanh$ bounds the conditioning scale to $(-1, 1)$, preventing unbounded perturbation during optimization.

Action Embedding Actions are represented as normalized planar displacements, $\bar{a}_{t-1} = a_{t-1}/a_{\max}$, where $a_{\max} = 0.4$ m. A two-layer MLP ϕ_a produces the action embedding $e_{a,t-1} = \phi_a(\bar{a}_{t-1})$. The resulting vector is broadcast across the K geometry tokens, yielding $E_{a,t-1} \in \mathbb{R}^{K\times d}$.

Selective Input The conditioning input for the SSM block then becomes the following:

$$X_t = [\text{LN}_g(Z_{g,t}), \text{LN}_h(H_{t-1}), E_{a,t-1}, E_{o,t}] \in \mathbb{R}^{K\times 4d}$$

Block A: Retention This component predicts a token-wise retention factor, determining how much of the hidden geometry state should be retained. We first predict an input-dependent step size:

$$\Delta_t = \min(1,\ \text{softplus}(W_\Delta X_t + b_\Delta - 2)) \in \mathbb{R}^d$$

The continuous-time diagonal decay is parametrized by

$$A = -\exp(a_{\log})$$

where $a_{\log} \in \mathbb{R}^d$ is a learned, time-independent parameter shared across tokens. Its input-dependent discretization is

$$\bar{A}_t = \exp(-\Delta_t \odot \exp(a_{\log})),$$

where $\odot$ indicates the element-wise multiplication. Therefore, the discretized retention $\bar{A}_t$ depends on the selective input vector X_t despite the continuous-time parameter $a_{\log}$ being static.

Block B: Candidate Input A candidate update is then constructed from the geometry, action, and visual-evidence embeddings. We construct a candidate content vector $I_t = [\text{LN}_g(Z_{g,t}), E_{a,t-1}, E_{o,t}]$. We then use the selective input vector X_t to gate the update:

$$B_t = \sigma(\text{Linear}_B(X_t)) \odot \tanh(\text{Linear}_u(I_t)),$$

where σ is ReLU. Note that linear transformations are applied to I_t and X_t to map them to the same dimensionality. While I_t does not contain the hidden state, it is gated by X_t, which does. Therefore, the previous hidden state controls how strongly the new geometry, action, and observation inputs are written into each channel.

Hidden State Update Blocks A and B are then used to calculate the hidden state update as follows:

$$H_t = A_t \odot H_{t-1} + (1 - A_t) \odot B_t$$

Block C: Bounded Geometry Correction The updated state is then passed through another gate to bound the geometry correction. Similar to block B, we calculate an update gated by X_t:

$$C_t = \text{sigmoid}(\text{Linear}_C(X_t)) \odot \tanh(\text{Linear}_R(\text{LN}(H_t)))$$

Residual Geometry Correction We finally correct the geometry token input by updating it residually using the C block:

$$\hat{Z}_{g,t} = Z_{g,t} + \gamma C_t$$

where $\gamma = 0.05$ is fixed, bounding residual updates to within $[-\gamma, \gamma]$. This prevents the dynamic corrector from overwriting the pretrained geometry representation. This residual head is zero-initialized, meaning the corrector initially implements $\hat{Z}_{g,t} = Z_{g,t}$ and learns to correct this gradually.

A.2.2 Geometry Encoder

To inject geometric information from this map into the latent space, we pre-train a scene-specific geometry encoder E_G and freeze it. It uses a multi-layer transformer architecture, with a final multi-head cross-attention layer where 16 input query tokens cross-attend to the point tokens to output a latent $Z_g \in \mathbb{R}^{16\times384}$, compatible with the DINOv2 latent dimension.

The encoder inputs 1024 sampled points in a fixed radius around the agent, with the yaw aligned with the world rather than the agent camera. This custom point cloud encoder enables the system to sample semantic point attributes on top of positions such as walkability and deformability.

The encoder is trained as part of a point cloud autoencoder, with a corresponding point cloud decoder D_G that initializes 100 learned queries, which self-attend and then cross-attend to the 16 encoder tokens to produce a 100-point reconstruction. This reconstruction target is trained using chamfer loss between the input and output point clouds, along with binary cross entropy losses for the walkability and deformability attributes of the nearest neighbor points assigned by the chamfer loss.

$$\mathcal{L}_{\text{reconstruction}} = \mathcal{L}_{\text{chamfer}} + \lambda_{\text{walk}}\mathcal{L}_{\text{walk}} + \lambda_{\text{deform}}\mathcal{L}_{\text{deform}}$$

A.2.3 Model Configuration

Vision Encoder RGB observations are resized to 224×224 and encoded using a frozen DINOv2 ViT-S/14 encoder, producing 256 patch tokens of dimension 384.

Geometry Encoder The frozen point-cloud encoder produces 16 geometry tokens of the same dimension as DINOv2. The geometry encoder and its corresponding autoencoder use two Transformer layers, four attention heads, and a hidden dimension of 384. The autoencoder is trained for 20 epochs using AdamW with a learning rate of 3×10^{-4}, weight decay 10^{-4}, and the reconstruction-loss weights $\lambda_{\text{walk}} = 0.1$ and $\lambda_{\text{deform}} = 0.01$. After pretraining, the decoder is discarded and the geometry encoder remains frozen.

SSM Geometry Corrector The SSM geometry corrector is implemented as specified in section A.2.1. It is trained jointly with the rest of the trainable components of the network.

Cross-Attention Fusion The 256 DINOv2 visual tokens act as queries, while the 16 corrected geometry tokens act as keys and values in a four-head cross-attention layer of width 384. The attention output is added residually to the visual tokens and followed by a pre-normalized feed-forward block with expansion ratio four and dropout 0.1. This preserves the visual token count and dimensionality, producing $Z_{f,t} \in \mathbb{R}^{256\times384}$, which is mean-pooled into the fused latent $z_{f,t} \in \mathbb{R}^{384}$. A separate EMA copy of the fusion module, updated with rate 0.999, constructs the stop-gradient prediction targets from the observation and uncorrected static-geometry tokens. The use of uncorrected geometry is motivated by Appendix A.9.

Action-Conditioned Predictor The predictor is implemented using four causal transformer blocks with four attention heads, feed-forward expansion ratio four, and dropout 0.1. The action is embedded by a two-layer MLP and injected into the attention and feed-forward portions of each block using adaptive layer normalization (AdaLN) (Xu et al., 2019). The input fused latent of dimension 384 is mean-pooled and projected down to a dimension of 256, and it is reprojected to 384 dimensions at the end.

Auxiliary Head A single auxiliary head A_ψ is shared between both encoded and predicted fused latents. It consists of a layer norm (LN) followed by a two-layer MLP with a hidden dimension of 256 and a GELU activation. The supervision target, $\bar{h}_t^{\text{EMA}}$, is a mean-pooled and layer-normalized stop-gradient EMA copy of the SSM hidden state, updated with rate 0.999. An auxiliary loss weight of $\lambda_A = 0.05$ is used.

Objective Training uses four-frame clips, corresponding to three consecutive action transitions. The fusion and predictor objectives are specified in section 3.5. For the encoded latent, gradients from the auxiliary loss update only the auxiliary head. For the predicted latent, the auxiliary-head parameters are stopped so that the loss instead structures the predictor and fusion blocks.

A.2.4 Training Sampling

Training clips are sampled offline from the cached observation, geometry, and action sequences. During aligned training, the canonical, matched-shifted-wall, and matched-static-obstacle conditions are sampled uniformly with repeats.

Variational fine-tuning uses a fixed deterministic cycle containing 90% aligned clips and 10% clips from the inconsistent benchmark conditions. The variational data covers all combinations of variance available in the benchmark, using a benchmark-matched distribution for the data. For each selected condition, the controllers are sampled in a round-robin fashion, then shuffled for sampling during training.

A.2.5 Two-Stage Training Procedure

We use a two-stage procedure to learn inconsistent behavior while limiting performance degradation in aligned environments.

Aligned Training In the first stage, each model is trained for 20 epochs using only aligned data. We use AdamW with a learning rate of 3×10^{-4}, weight decay 10^{-4}, and batch size 64. The checkpoint with the best validation loss over all 20 epochs is selected for further finetuning.

Variational Finetuning The selected aligned checkpoint is then fine-tuned for 2 epochs using the 90/10 aligned-variation mixture. The learning rate is reduced to 3×10^{-5}, while the batch size, number of updates, weight decay, and EMA rates remain unchanged. This configuration prevents latent collapse while exposing the model to inconsistent geometry, deformable interactions, and corrupted observations.

A frozen copy of the selected aligned checkpoint is additionally used as a teacher during fine-tuning. The teacher loss is applied only to aligned samples:

$$\mathcal{L}_{\text{teacher}} = \lambda_e ||z^S_{\text{fused}} - z^T_{\text{fused}}||_2^2 + \lambda_p ||\hat{z}^S_{\text{fused}} - \hat{z}^T_{\text{fused}}||_2^2 + 0.1||H^S - H^T||_2^2, \qquad \mathcal{L} = \mathcal{L}_{\text{WM}} + \mathcal{L}_{\text{teacher}}$$

Where $\lambda_e = 1.0, \lambda_p = 1.0$.

A.2.6 Planning Protocol

Identical to DINO-WM, we use the Cross-Entropy Method for planning. It optimizes an 8-step action sequence from the current latent state. We use 256 candidate action sequences, retain 25 elites per iteration, and run 5 iterations total. Actions are sampled from a Gaussian distribution with an initial action standard deviation of $0.15\,\text{m}$, and a minimum standard deviation of $0.01\,\text{m}$. These are projected onto the valid $||a_t||_2 \leq 0.40\text{m}$ action disk. The final plan is the mean of the final elites, and the first action from this plan is selected to be executed in the environment. The goal objective is

a fused latent z_{goal} formed by fusing the RGB latent from the goal observation and geometry latent from the local static-map point cloud centered at the goal using the cross-attention fusion module.

A.2.7 Inference Cost and Latency

We measure inference on a single NVIDIA A6000 with an environment batch size of 1. Rendering and simulation using the TDW environment are excluded from all measurements.

Loaded Parameters Total number of parameters in GPU memory during action selection, including frozen components used for inference.

Latency per Action We measure the wall-clock time from an observation input to an action output, including all encoding, prediction, and planning steps. Both median and 95-th percentile (p95) measurements are reported.

FLOPs per Action Furthermore, we report the number of floating point operations required to produce one action from one observation, measured for all the same processes as latency per action.

Model Evaluations per Action Number of model evaluations required to produce an action. For CEM, this is calculated as a product of the number of iterations, number of candidate action sequences, and the rollout horizon. Our configuration uses 5 iterations, 256 candidates, and a horizon length of 8, resulting in 10240 transition predictions per action. For methods without CEM, we report the model's native workload.

Peak Inference Memory Maximum allocated GPU memory while selecting an action.

A.2.8 Training Data and Compute

Another axis on which we evaluate our models against baselines is the amount of training data and compute required to produce the reported results.

Collection Mode Indicates whether training uses a fixed offline dataset or requires online data collection with the environment.

Unique Transitions Number of distinct environment transitions or clips available during training, not counting reuse across epochs. This indicates how much raw data is needed to train the model.

Transition Presentations Total number of temporal transitions presented to the optimizer, which includes repeated sampling. This distinguishes raw data requirements from repeated optimization over said samples.

Optimizer Updates Number of gradient updates performed during training.

Trainable Parameters Total number of parameters updated during training, which excludes any frozen components.

Representation GPU-Hours Computation required to train the encoder, latent dynamics, and predictor.

Controller GPU-Hours Additional computation required to train the controller for planning. This is also zero for CEM planners.

Total GPU-Hours Total compute required to train the entire model.

All training costs are normalized to a single NVIDIA A6000. The geometry encoder for our model and many baseline variations is trained once and reused across methods, but as it is trained on the particular environment, its compute is added to the representation and total GPU-Hours metrics.

A.2.9 LATENT DIAGNOSTIC PROTOCOL

The protocol for various latent diagnostics is controlled as much as possible. Unless otherwise stated, all token representations are mean-pooled to yield 384-dimensional vectors. Centered linear CKA is used to compare representation spaces, and MSE between latents is averaged over samples and feature dimensions.

Linear Probes All reported linear probes are ridge regressors with regularization coefficient 1.0 and fitted on frozen models. For Table 4, we use five-fold cross validation over the 30 held-out evaluation layouts. The targets are physical wall center relative to the agent, $(X_{\text{phys}} - X_{\text{agent}}, Z_{\text{phys}} - Z_{\text{agent}})$, and the distance between the physical wall and supplied map, $(X_{\text{phys}} - X_{\text{map}}, Z_{\text{phys}} - Z_{\text{map}})$. The reported value is the Pearson correlation between the probe predictions and ground truth targets.

For Table 22, probes are trained on 20 layouts and evaluated on 10 held-out layouts. Nine X and Z targets are used:

- agent X and Z,
- goal X and Z,
- goal distance,
- wall center X and Z,
- and the X coordinates of the wall's left and right edges.

At each prediction horizon, a separate probe is fitted on the corresponding autoregressively predicted fused latent. The reported values are averaged over all nine targets and ten environment conditions.

Representation Statistics To calculate the effective rank $\text{erank}(Z)$ of any latent Z with covariance eigenvalues $\{\lambda_j\}_{j=1}^{d}$, we define

$$p_j = \frac{\lambda_j}{\sum_k \lambda_k}, \qquad \text{erank}(Z) = \exp\left(-\sum_j p_j \log p_j\right)$$

Mean feature standard deviation is the average standard deviation over the 384 feature dimensions for any latent. In table 15, each of the 16 geometry tokens is treated as an observation for Z_g, while Z_{fused} is mean-pooled over its 256 tokens. Tables 13 and 21 use 2500 four-frame clips balanced across environment conditions from the validation split.

Geometry Perturbation For Table 13, corrected geometry tokens are permuted across clips in each minibatch, with fixed visual tokens, temporal positions, and actions. For permutation π, we compute

$$\widetilde{z}_{f,t} = \text{Fusion}(Z_{o,t}, \hat{Z}^{\pi}_{g,t}),$$

and report the mean relative changes as follows:

$$\Delta_{\text{fused}} = \mathbb{E}\left[\frac{\|\widetilde{z}_{f,t} - z_{f,t}\|_2}{\|z_{f,t}\|_2}\right], \qquad \Delta_{\text{pred}} = \mathbb{E}\left[\frac{\|P(\widetilde{z}_f, a) - P(z_f, a)\|_2}{\|P(z_f, a)\|_2}\right].$$

This only changes the geometry tokens input into the fusion module, thus measuring how strongly geometry affects the fused representation and prediction.

A.3 Latent World Model Baselines

We evaluate our model against a variety of latent world model baselines such as LeWM, PLDM, and DINO-WM, which are the most direct comparisons to GeoMamba-WM. They all operate with similar latent state predictors, and share identical action spaces, plan using CEM with the same latent-distance objective, and are all trained using the same data and curriculum. By controlling for these factors and using CEM as an offline planner, the representation quality of the models is isolated and can be compared directly. Using learned planners like in Appendix A.4 can blur these representational differences and require additional supervision such as reward functions and online data collection.

A.3.1 LeWM

LeWorldModel (Maes et al., 2026b) is a jointly-trained latent world model framework that trains the vision encoder along with the predictor. It uses the Joint Embedding Predictive Architecture (JEPA) framework, therefore being a lightweight world model option, and implements a SIGReg, an additional regularization loss to prevent latent space collapse. We use the implementation provided by `stable-worldmodel` (Maes et al., 2026a), retaining its prediction objective, SIGReg loss, and jointly trained visual encoder.

Our RGB-only implementation uses a ViT encoder with 192-dimensional patch tokens and the standard LeWM predictor. We additionally evaluate a geometry-conditioned variant using our frozen geometry encoder. Its 384-dimensional tokens are projected to the 192-dimensional LeWM token width, after which a similar four-head fusion block to our model is applied to obtain a fused latent. These are pooled and passed to the unchanged LeWM predictor. Gradients update the visual encoder, fusion model, and predictor, while the geometry encoder remains frozen. At inference time, LeWM is evaluated using the benchmark's locked CEM protocol and terminal latent-distance objective rather than LeWM-specific planner settings.

A.3.2 PLDM

Planning with Latent Dynamics Models (Sobal et al., 2025), similar to LeWM, jointly trains its encoder and predictor. Rather than using SIGReg, it uses a variety of variance, covariance, temporal-variance, and temporal-alignment regularizers to prevent latent space collapse. We use the standardized `stable-worldmodel` implementation and the default loss coefficients.

For a controlled comparison with LeWM, PLDM uses the same visual encoder, predictor size, input resolution, training data, and planning interface. We also evaluate a geometry-conditioned model using the same frozen geometry encoder, along with the token projection module used for LeWM described above. At inference time, PLDM is evaluated as per the benchmark's CEM protocol rather than its own planner settings.

A.3.3 DINO-WM

GeoMamba-WM is built upon DINO-WM, due to which it is the primary baseline. For the baseline comparison, we use the exact DINO-WM specifications by using a frozen DINOv2 encoder and directly predicting spatial tokens as opposed to a mean-pooled latent. We also evaluate a geometry-conditioned model, which is reported as the No-SSM baseline in various ablations.

A.3.4 Results

Performance As discussed in section 6.1, GeoMamba-WM outperforms all other methods in inconsistent scenarios, while training with aligned-data only yields the best results in aligned scenes.

Training Cost All latent world models use the same 11048 unique training transitions and 3322 optimizer updates. GeoMamba-WM consumes half the transition presentations, equal to DINO-WM with geometry, and fewer than the RGB-only baselines. Although the SSM and auxiliary components expand the trainable parameters beyond DINO-WM, its measured training time is lower.

Inference Cost Despite increasing the loaded parameters to above 40 million, it retains nearly the same inference cost as DINO-WM and requires similar GFLOPs per action. It also remains significantly less expensive than LeWM and PLDM with respect to latency and compute, though VRAM usage of GeoMamba-WM is the highest across all baselines.

Table 6: Success rates and SoftSPL in percentages for latent world models. Values are mean ± standard deviation over three evaluation seeds.

Method	Aligned	Inconsistent	Geometry	Deformation	Noise
Success rate (%)					
PLDM	0.00±0.00	0.18±0.31	0.00±0.00	0.00±0.00	0.23±0.40
PLDM + Geometry	0.00±0.00	3.90±0.81	4.38±0.63	2.43±0.60	2.31±0.40
LeWM	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00
LeWM + Geometry	0.00±0.00	2.13±0.00	1.88±0.63	4.17±0.00	2.55±0.40
DINO-WM	4.44±3.85	2.48±1.11	2.29±1.30	2.43±1.59	2.78±0.70
DINO-WM + Geometry	33.33±3.33	26.95±2.40	25.42±3.21	28.12±3.12	23.38±2.00
GeoMamba-WM (Aligned)	**55.56±11.71**	27.31±0.61	25.00±0.00	32.29±2.08	25.00±0.69
GeoMamba-WM	37.78±6.94	**37.06±1.71**	**33.75±0.63**	**43.06±1.20**	**36.34±4.07**
SoftSPL (%)					
PLDM	3.95±0.51	0.69±0.25	0.70±0.25	0.04±0.03	0.61±0.41
PLDM + Geometry	5.99±0.21	3.75±0.15	3.89±0.03	1.45±0.14	3.21±0.12
LeWM	0.67±0.38	0.15±0.01	0.13±0.04	0.00±0.01	0.14±0.02
LeWM + Geometry	1.38±0.67	2.51±0.20	2.34±0.43	2.79±0.28	2.67±0.26
DINO-WM	9.00±2.20	6.14±0.55	5.98±0.61	6.28±0.54	5.31±0.47
DINO-WM + Geometry	29.14±1.66	17.32±0.91	17.23±0.97	14.50±1.32	16.08±0.94
GeoMamba-WM (Aligned)	**36.79±4.82**	14.71±0.49	14.18±0.29	13.43±0.31	13.47±0.76
GeoMamba-WM	33.03±2.20	**18.84±0.63**	**17.94±0.48**	**17.25±0.86**	**18.39±0.48**

Table 7: Task-specific training cost of the latent world models.

Method	Precision	Transition Presentations	Trainable Params	A6000 GPU-h
LeWM	BF16	1,275,648	18.03M	1.83
LeWM + Geometry	BF16	1,275,648	18.55M	1.74
PLDM	BF16	1,275,648	18.03M	1.82
PLDM + Geometry	BF16	1,275,648	18.55M	1.75
DINO-WM	FP32	850,432	5.38M	0.36
DINO-WM + Geometry	FP32	637,824	6.36M	0.27
GeoMamba-WM	FP32	637,824	10.99M	0.23

Table 8: Inference cost of latent world models under the shared CEM protocol, measured end-to-end per executed action on an NVIDIA A6000. All methods use 256 candidates, 25 elites, 5 CEM iterations, and a rollout horizon of 8.

Method	Precision	Loaded Params	Median Latency	P95 Latency	GFLOPs/Action	Peak VRAM
LeWM	BF16	18.03M	384.4 ms	398.1 ms	724.9	154.6 MiB
LeWM + Geometry	BF16	22.71M	400.1 ms	410.8 ms	736.2	173.4 MiB
PLDM	BF16	18.03M	377.5 ms	397.5 ms	724.9	154.6 MiB
PLDM + Geometry	BF16	22.71M	400.2 ms	406.0 ms	736.2	173.4 MiB
DINO-WM	FP32	27.44M	199.8 ms	205.6 ms	287.3	249.2 MiB
DINO-WM + Geometry	FP32	34.34M	200.0 ms	225.5 ms	299.2	276.0 MiB
GeoMamba-WM	FP32	43.41M	200.1 ms	203.8 ms	299.4	310.7 MiB

A.4 End-to-End Comparisons

A.4.1 NeDreamer

NeDreamer (Bredis et al., 2026) is a model-based reinforcement learning method based on Dreamer that removes the decoder to reduce planning-time latency. An image encoder first embeds the RGB observation, after which a recurrent state space model (RSSM) updates and maintains a latent state based on previous actions. Instead of pixel reconstruction, a causal temporal transformer is used to predict the next latent with a redundancy-reduced alignment objective. Additionally, it learns reward and continuation predictors to train actor and critic models on imagined latent rollouts in an online fashion. At inference time, the actor is able to directly map an RGB observation to an action without online trajectory optimization. We use NeDreamer-12M for our baseline.

Adaptation to CEM To give NeDreamer a fair comparison to our model, we first adapt the model to a CEM-based planner and offline learning protocol. We retain the encoder, RSSM, and temporal transformer, but remove the decoder, rewards and continuation heads, and the actor and critic models. The remaining RGB-only model is trained using the exact aligned and variational curriculum as GeoMamba-WM. For planning with CEM, each candidate action sequence is propagated through the adapted NeDreamer model to extract final latents. These are compared to the embedded goal observation as the CEM objective. We retain the common configuration of 256 candidates, 25 elites, 5 iterations, and a rollout horizon of 8.

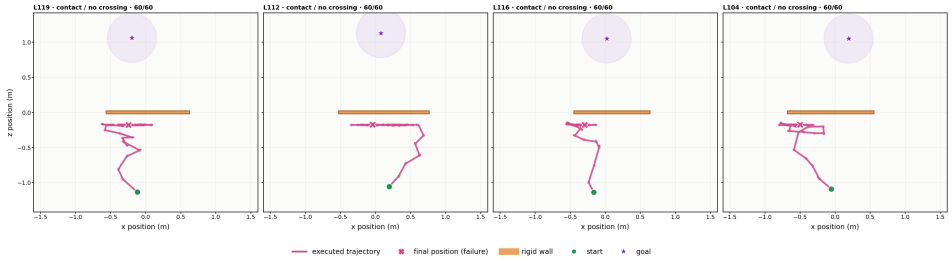

Figure 8: Sample planning failures when adapting NeDreamer to use a CEM planner.

Initial validation on aligned environments demonstrates the failure of the model to learn proper planning behavior. Figure 8 shows four sample layouts of the model only learning to go forward, with a full aligned evaluation success rate of only 10%. This error indicates that NeDreamer cannot be simply adapted to a CEM-based planner, necessitating full model training for appropriate comparison.

NeDreamer Actor-Critic Baseline For the reported NeDreamer baseline, we retain the method's native model-based RL planner through the reward and continuation heads, and actor and critic models. The model is not modified to receive the geometric prior, operating only on 64×64 RGB observations. Its two-dimensional policy output lies in $[-1, 1]^2$, which is rescaled to the benchmark's $0.4\,\mathrm{m}$ maximum displacement per action. Training layouts are sampled with an equivalent distribution to the benchmark training set. To appropriately train this model, we also define a privileged reward function r_t:

$$r_t = \frac{d_{t-1} - d_t}{0.4} + 5 \times \mathbb{1}[\text{first success at } t] - 0.01$$

where d_t is the Euclidean distance to the goal. The function rewards reduced distance towards the goal, greatly rewards reaching the goal, and penalizes stalling through the 0.01 time penalty. The actor and critic are optimized using the original NeDreamer formulation of a 15-step latent imagination. During inference, the deterministic actor simply selected one action step without any planning-time trajectory optimization.

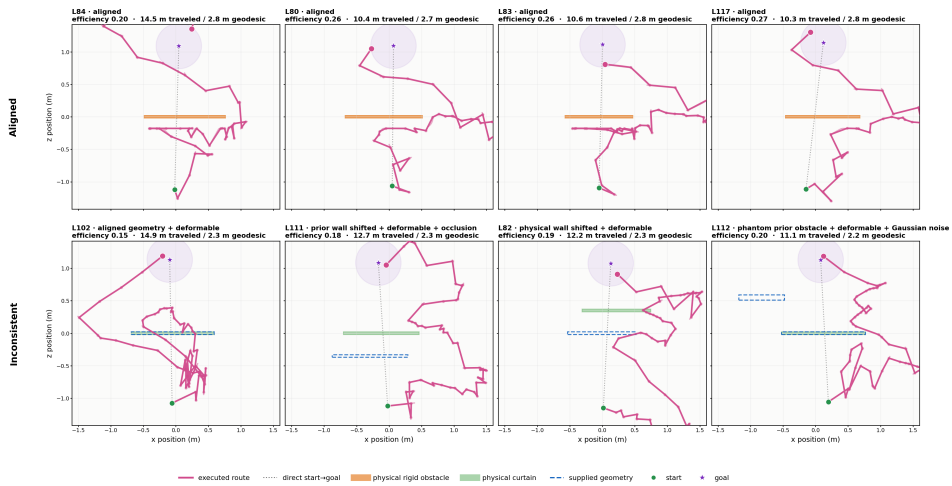

Figure 9: Sample paths taken by NeDreamer using its native actor-critic planner in various scenarios.

A.4.2 3D-BELIEF

3D-Belief maintains an online 3D scene and uses a frontier-based navigation controller instead of CEM.

belief update → occupancy update → frontier sampling → view imagination → A* planning

We make several task-level adaptations to make the model work with GOI-NAV. The original model uses semantic goal scoring, which is replaced with DINOv3 (Siméoni et al., 2026) feature similarity between the imagined view and the RGB goal observation. The controller samples three frontier candidates and imagines up to three keyframes for each candidate using five denoising steps. The selected frontier is passed to the A* planner, and its first waypoint is then bounded to the benchmark's $0.4\,\mathrm{m}$ maximum displacement per action. We use simulator odometry to retain a selected frontier across consecutive actions and only replan when the agent becomes stuck. Since GOI-NAV provides a geometric prior, it is rasterized into the controller's occupancy map with deformable obstacles labeled accordingly. These changes preserve the method's belief, imagination, frontier-selection, and path-planning pipeline, with the only changes being the goal objective, geometric prior injection, and action output projection.

A.4.3 PERSIST

PERSIST (Garcin et al., 2026) is a generative video world model that maintains an explicit dynamic voxel representation of the environment. Given an initial observation, optionally an initial voxel state w_0, and a conditioning action, the model autoregressively predicts the next voxel latent using a voxel diffusion transformer and VAE, uses a causal camera transformer to predict the camera state, and finally generates the next visual observation using another pixel diffusion transformer and VAE. While this explicit 3D voxel state allows PERSIST to generate coherent autoregressive long-horizon rollouts, it is computationally expensive and not designed for real-time robot planning. However, due to its geometry-conditioned world model formulation with a persistent state, we compare it as a relevant baseline.

Adaptation to CEM Our standard CEM controller evaluates 10240 world-model transitions for each executed action. While this works for lightweight latent world models, running 10240 voxel and image transitions per environment step is computationally prohibitive. A single PERSIST-S transition and controller decision on an NVIDIA A6000 GPU requires 292.2 ms and 4.00 TFLOPs. For a single action, this would require approximately 50 minutes and 40 PFLOPs. We report this as a linear projection of the measured single-transition values rather than executing a full PERSIST-S evaluation with CEM.

PERSIST-S We instead use the smaller PERSIST-S configuration, initialized with a voxelized benchmark-provided geometric prior w_0. While the original model is autoregressive, we perform planning in a closed-loop fashion. Following each executed action, the frozen voxel diffusion predicts the next voxel state using two denoising steps, while the corresponding pixel-level observation is provided by the environment, allowing us to bypass the pixel diffusion step. The output features include a spatially pooled voxel latent, a pooled pixel latent, and a predicted camera representation from the causal camera transformer, thus producing a 1498-dimensional state vector. All PERSIST components remain frozen while training the controller.

SAC Controller We train a Soft Actor-Critic (SAC) planner on the frozen PERSIST representations. The actor is a three-layer MLP with hidden width 256, LayerNorm, and SiLU activations, and outputs a two-dimensional action bounded to the benchmark's maximum displacement. Two independent critics estimate the action value, with target critics updated as an average of multiple training iterations. The controller uses the same distance-based reward as NeDreamer. During planning, the deterministic actor model uses a single PERSIST-S transition for an action, avoiding expensive online trajectory optimization.

A.4.4 GeoMamba-WM With Soft Actor-Critic

A CEM controller is typically used with latent world models because the offline planner isolates the quality of the learned representations and dynamics. However, when compared with end-to-end model-based RL, it results in an unfair comparison due to its latency and lack of supervision. Therefore, to provide a strict controller-matched evaluation against PERSIST-S and NeDreamer, we similarly trained an identical SAC controller to the one used with PERSIST-S on top of the frozen GeoMamba-WM backbone. By using the same online collection protocol, total transitions, and distance-based reward function used for NeDreamer and PERSIST, the actor learns to map GeoMamba-WM's fused latents directly to continuous planar displacements, bypassing the need for CEM trajectory optimization.

A.4.5 Results

Table 9: End-to-end model performance comparison. Values report mean ± standard deviation over three evaluation seeds, in percentages.

Method	Aligned	Inconsistent	Geometry Marginal	Deformation Marginal	Noise Marginal
Success Rate (%)					
GeoMamba-WM	37.78 ± 6.94	37.06 ± 1.71	33.75 ± 0.63	43.06 ± 1.20	36.34 ± 4.07
NeDreamer (Native Actor-Critic)	88.89 ± 3.85	68.79 ± 0.31	68.13 ± 1.65	60.07 ± 3.94	66.90 ± 2.12
3D-Belief	86.67 ± 6.67	40.78 ± 2.21	41.67 ± 2.19	15.62 ± 3.76	40.51 ± 2.63
PERSIST-S + SAC	**100.00 ± 0.00**	**94.41 ± 1.88**	**94.38 ± 1.77**	**94.27 ± 3.68**	**92.71 ± 2.46**
GeoMamba-WM + SAC	**100.00 ± 0.00**	86.35 ± 2.62	87.29 ± 2.37	79.86 ± 2.17	82.18 ± 3.43
SoftSPL (%)					
GeoMamba-WM	33.03 ± 2.20	18.84 ± 0.63	17.94 ± 0.48	17.25 ± 0.86	18.39 ± 0.48
NeDreamer (Native Actor-Critic)	25.86 ± 1.68	22.16 ± 0.97	21.84 ± 0.77	19.08 ± 1.43	22.17 ± 0.86
3D-Belief	51.90 ± 4.69	26.76 ± 0.98	27.58 ± 1.33	12.15 ± 1.28	26.43 ± 1.28
PERSIST-S + SAC	81.99 ± 0.35	**56.28 ± 1.08**	**55.75 ± 0.35**	**46.79 ± 0.81**	**54.63 ± 1.11**
GeoMamba-WM + SAC	**83.42 ± 0.38**	47.84 ± 0.55	48.15 ± 0.35	28.54 ± 0.72	44.86 ± 0.83

Table 10: Task-specific training cost of end-to-end models.

Method	Supervision	Unique Transitions	Updates	Transition Presentations	Trainable Params	A6000 GPU-h
GeoMamba-WM	Offline predictive	11,048	3,322	637,824	10.99M	0.23
NeDreamer	Online RL + reward	25,007	9,994	10,073,952	14.08M	6.56
3D-Belief	Offline RL + reward	4,977	2,001	10,005	467.09M	1.01
PERSIST-S + SAC	Online RL + reward	25,007	20,000	5,121,792	1.55M	15.4
GeoMamba-WM + SAC	Offline + online RL + reward	25,007	20,000	5,757,824	12.54M	5.7

Table 11: End-to-end inference cost measured per executed action on an NVIDIA A6000.

Method	Controller	Decision Computation	Loaded Params	Median Latency	P95 Latency	GFLOPs/Action	Peak VRAM
GeoMamba-WM	CEM	10,240 WM transitions	43.41M	200.1 ms	203.8 ms	299.4	310.7 MiB
NeDreamer	Actor-Critic	1 actor forward pass	25.00M	3.5 ms	3.6 ms	0.163	91.1 MiB
3D-Belief	Belief + A*	Belief update + 9 views	717.76M	22.6 s	22.8 s	127,567.9	28,053.9 MiB
PERSIST-S + SAC	SAC	1 voxel update + actor	1,286.51M	292.2 ms	309.7 ms	4,001.1	4,796.6 MiB
GeoMamba-WM + SAC	SAC	1 latent-state update + actor	43.93M	17.0 ms	29.0 ms	24.41	237.0 MiB

Performance When comparing all the end-to-end models, the use of task-specific actor-critic planners trained with distance-based rewards and online RL provides massive performance gains over offline, reward-free CEM. NeDreamer achieves significantly higher success rates than GeoMamba-WM (CEM) in all scenarios, though the low SoftSPL is indicative of poor path efficiency. This can be seen in Figure 9, where NeDreamer takes indirect winding paths to reach the goal, resulting in an aligned SoftSPL even lower than GeoMamba-WM. This can be owed to the lack of geometry grounding in NeDreamer, indicating the utility of providing world models with a geometric prior.

3D-Belief does use geometry grounding and maintains an online 3D scene and uses occupancy maps to perform A* planning, resulting in stronger success rates than GeoMamba-WM with CEM in most scenarios, and higher SoftSPL than NeDreamer in most. It should be noted that the deformation success rates and SoftSPL values are the lowest for 3D-Belief, revealing that explicit occupancy assumptions do not translate well to deformable object traversal even when given explicit deformability labels.

Ultimately, due to its dynamic voxel representation, PERSIST-S with the custom SAC controller achieves state-of-the-art results on this environment, with the highest success rates across all scenes. However, evaluating the controller-matched GeoMamba-WM with the SAC planner reveals that our model maintains competitive performance, with the highest aligned SoftSPL and 100% success rate, and the second highest performance on all inconsistent tasks. This demonstrates that geometry-conditioned latent representations are richer than the standard RGB-only latents of NeDreamer when provided with identical RL advantages, and also that geometry-observation inconsistencies can be solved entirely in the latent space without requiring the explicit 3D representation of PERSIST or 3D-Belief.

Training Cost The cost of training varies depending on the required supervision of the planning method. GeoMamba-WM with a CEM planner is particularly cheap, requiring only 0.23 GPU Hours for training. 3D-Belief uses the least raw data, but requires more training time due to the heavier architecture.

All models utilizing online RL are trained in a data-matched fashion, using 25000 environment interactions. NeDreamer and GeoMamba-WM both have few trainable parameters and thus require less training time, while PERSIST-S has the least number of parameters but requires significantly more training time due to the frozen voxel diffusion executed with each iteration.

Inference Cost Due to NeDreamer's use of a deterministic actor policy, it only requires one forward pass per action. This results in an unparalleled inference latency of 3.5 ms. Conversely, generative models like PERSIST-S require two voxel denoising steps per action, taking 292.2 ms of latency per action, and 3D-Belief's multiple diffusion steps render it unusable in real-time with a 22.6 s per-action latency.

While our offline CEM planner requires 200.1 ms to evaluate all 10240 predicted transitions, using an SAC controller with GeoMamba-WM reduces the inference latency to just 17.0 ms and 24.4 GFLOPs. This makes it over $17\times$ faster and $164\times$ more compute-efficient than PERSIST-S with the same SAC controller, achieving near-SOTA performance while remaining within the bounds of real-time control.

A.5 GEOMETRY ABLATIONS

We perform a series of ablations on the geometry encoder and input to justify our model decisions.

A.5.1 FUSION MECHANISM

GeoMamba-WM uses cross attention to fuse the corrected geometry tokens with the visual tokens. We ablate the fusion mechanism to justify the use of cross attention for this purpose.

For AdaLN, we first mean-pool the geometry tokens and map that to scale and shift parameters using a zero-initialized MLP that modulates every normalized observation token. For concatenation, we also mean-pool the geometry tokens. These are repeated and concatenated with every observation token, then reprojected to the token dimension of 384. This projection is also initialized as $[I\ 0]$, preserving the observation representation at initialization.

Table 12: Comparison between fusion mechanisms. Values report mean $\pm$ standard deviation over three evaluation seeds, in percentages.

Method	Aligned	Inconsistent	Geometry Marginal	Deformation Marginal	Noise Marginal
Success Rate (%)					
Cross Attention (Ours)	**37.78 ± 6.94**	**37.06 ± 1.71**	**33.75 ± 0.63**	**43.06 ± 1.20**	**36.34 ± 4.07**
Adaptive Layer Norm	7.78 ± 6.94	0.71 ± 0.30	0.62 ± 0.00	0.00 ± 0.00	0.69 ± 0.70
Concatenation	28.89 ± 1.92	12.77 ± 0.92	11.67 ± 1.57	11.11 ± 3.35	13.19 ± 0.70
SoftSPL (%)					
Cross Attention (Ours)	**33.03 ± 2.20**	**18.84 ± 0.63**	**17.94 ± 0.48**	**17.25 ± 0.86**	**18.39 ± 0.48**
Adaptive Layer Norm	14.10 ± 2.65	6.48 ± 0.37	6.51 ± 0.40	5.23 ± 0.32	5.98 ± 0.37
Concatenation	25.28 ± 2.03	13.41 ± 0.36	12.98 ± 0.60	10.99 ± 1.46	13.10 ± 0.14

Results in table 12 demonstrate the effectiveness of cross-attention as a fusion mechanism, achieving higher success rates and SoftSPL across all scenarios. CKA similarity values in table 13 show that the z_{fused} latent is more similar to the corrected geometry latent $\hat{z}_g$ for both AdaLN and concatenation. However, Δ_{fused} and Δ_{pred} denote the shifts in the fused and predicted latents, respectively, from perturbing $\hat{z}_g$, and they show that cross attention has the largest effect on both the fused representation and prediction. This indicates a stronger propagation of geometric information into the latent space using cross attention, supporting the higher success rates.

Table 13: Comparison of latent space diagnostics for fusion mechanisms.

Method	CKA(z_o, z_{fused})	CKA$(\hat{z}_g, z_{\text{fused}})$	Δ_{fused}	Δ_{pred}
Cross Attention (Ours)	0.8809	0.2360	**0.3512**	**0.2703**
Adaptive Layer Norm	**0.9354**	0.2489	0.3262	0.1183
Concatenation	0.8535	**0.3069**	0.3126	0.2279

A.5.2 GEOMETRY ENCODER

We also ablate the use of the frozen geometry encoder by training a model with a randomly initialized geometry encoder that is jointly trained using the world-model objective, similar to LeWM. We ablate this without the use of the SSM or auxiliary heads to isolate the effect of the geometry encoder.

Results show poor performance with joint training, indicating the instability of training without a fixed encoded geometry latent space. This is further revealed by comparing the Z_g and z_{fused} latents. We evaluate latents from 2500 validation clips balanced across all conditions and report the standard deviation and effective ranks of the corresponding latents in table 15. We find that the frozen encoder captures a richer geometry representation than the jointly trained encoder, which then results in a better fused latent representation.

Table 14: Comparison of geometry encoder implementations without an SSM module or auxiliary heads. Values report mean ± standard deviation over three evaluation seeds, in percentages.

Method	Aligned	Inconsistent	Geometry Marginal	Deformation Marginal	Noise Marginal
Success Rate (%)					
Frozen Encoder	**33.33 ± 3.33**	**26.95 ± 2.40**	**25.42 ± 3.21**	**28.12 ± 3.12**	**23.38 ± 2.00**
Jointly Trained Encoder	4.44 ± 5.09	8.16 ± 0.81	7.08 ± 0.36	9.72 ± 2.17	8.80 ± 1.06
SoftSPL (%)					
Frozen Encoder	**29.14 ± 1.66**	**17.32 ± 0.91**	**17.23 ± 0.97**	**14.50 ± 1.32**	**16.08 ± 0.94**
Jointly Trained Encoder	16.35 ± 0.78	13.17 ± 0.93	12.89 ± 0.69	10.01 ± 0.85	13.15 ± 0.66

Table 15: Representation statistics for frozen and jointly trained geometry encoders.

Metric	**Frozen Encoder**	Jointly Trained Encoder
Z_g Mean Feature Std.	**0.793**	0.194
Z_g Effective Rank	**4.62**	1.87
z_{fused} Mean Feature Std.	**1.348**	1.324
z_{fused} Effective Rank	**38.81**	37.40

A.5.3 Geometry Input

Our model samples a 1 meter radius around the agent to extract 1024 points and passes that point cloud to the frozen geometry encoder. However, baselines like 3D-Belief operate on the entire map. We therefore ablate our geometry input by training two new geometry encoders and corresponding models. The first is given the full map within the world coordinate frame. The second is given the full map translated to an agent-centered coordinate frame, such that the agent's current location is set to $(0, 0)$. This isolates the effect of passing in global information without removing the localization benefits of sampling the prior relative to the agent coordinate.

Table 16: Comparison of geometry input ablations on GeoMamba-WM in the GOI-NAV benchmark. Values report mean ± standard deviation over three evaluation seeds, in percentages.

Method	Aligned	Inconsistent	Geometry Marginal	Deformation Marginal	Noise Marginal
Success Rate (%)					
World-Coordinate Full Map	3.33 ± 3.33	1.06 ± 0.00	1.04 ± 0.36	0.00 ± 0.00	1.16 ± 0.40
Agent-Centered Full Map	35.56 ± 9.62	7.09 ± 2.15	7.08 ± 3.08	4.17 ± 1.80	5.32 ± 1.06
Ours	**37.78 ± 6.94**	**37.06 ± 1.71**	**33.75 ± 0.63**	**43.06 ± 1.20**	**36.34 ± 4.07**
SoftSPL (%)					
World-Coordinate Full Map	18.97 ± 0.69	8.83 ± 0.14	8.82 ± 0.22	5.14 ± 0.31	8.06 ± 0.17
Agent-Centered Full Map	25.42 ± 4.06	8.55 ± 0.74	8.61 ± 1.20	5.37 ± 0.60	7.35 ± 0.60
Ours	**33.03 ± 2.20**	**18.84 ± 0.63**	**17.94 ± 0.48**	**17.25 ± 0.86**	**18.39 ± 0.48**

Results show that our smaller sampled map outperforms both ablations, indicating that providing the global map information is inefficient for agent planning, which is primarily concerned with the local surroundings of the agent. The results for the world-coordinate full map are nearly as poor as the RGB-only DINO-WM, showing the geometric signal is unable to help with planning. Additionally, performance for both ablations is particularly poor in inconsistent scenarios, showing that the ability of the SSM to resolve inconsistencies between the observation and geometry is also impeded by the extra global geometric information. Translating the global map to an agent-centered coordinate system improves success rates across the board, and particularly so for aligned scenarios, showing that the supplied map also aids the model and planning with agent localization.

A.5.4 Odometry Drift

Our model relies on accurate odometry to sample a local point cloud from the geometric prior. In real-world scenarios, odometry is rarely accurate. Therefore, we introduce drift into the geometry sampling procedure to evaluate how robust the SSM is to this inconsistency.

We simulate drift by sampling Gaussian noise ϵ and adding it to the position p_t at each timestep, clipping the odometry at a maximum radius r of 0.1m and 0.4m for each ablation.

$$\bar{p}_t = p_t + \text{clip}_r(\delta_{t-1} + \epsilon_t), \qquad \epsilon_t \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$$

Furthermore, we train models with odometry drift applied to the training data to assess whether this improves the models' robustness to drift.

Table 17: Application of odometry drift when inferencing and training GeoMamba-WM on the GOI-NAV benchmark. Values report mean ± standard deviation over three evaluation seeds, in percentages.

Method	Aligned	Inconsistent	Geometry Marginal	Deformation Marginal	Noise Marginal
Success Rate (%)					
Ours	37.78 ± 6.94	**37.06 ± 1.71**	**33.75 ± 0.63**	**43.06 ± 1.20**	36.34 ± 4.07
Ours + Noisy Inference (0.1m)	36.67 ± 3.33	36.17 ± 1.41	32.29 ± 2.19	39.24 ± 3.66	**36.65 ± 1.75**
Noisy Training + Inference (0.1m)	**38.89 ± 1.92**	36.88 ± 0.81	32.71 ± 1.57	42.36 ± 1.59	35.80 ± 1.45
Ours + Noisy Inference (0.4m)	35.56 ± 5.10	35.12 ± 4.33	31.17 ± 2.89	37.63 ± 5.74	34.57 ± 4.62
Noisy Training + Inference (0.4m)	34.44 ± 6.94	34.77 ± 1.06	32.05 ± 1.44	38.44 ± 4.54	33.19 ± 0.80
SoftSPL (%)					
Ours	**33.03 ± 2.20**	**18.84 ± 0.63**	17.94 ± 0.48	**17.25 ± 0.86**	**18.39 ± 0.48**
Ours + Noisy Inference (0.1m)	32.51 ± 1.05	18.12 ± 0.14	18.16 ± 0.47	17.03 ± 0.62	18.21 ± 0.34
Noisy Training + Inference (0.1m)	32.78 ± 1.58	18.70 ± 0.54	**18.33 ± 0.68**	17.20 ± 0.93	18.30 ± 0.40
Ours + Noisy Inference (0.4m)	31.77 ± 3.11	17.33 ± 0.78	17.52 ± 0.66	16.15 ± 0.65	17.58 ± 0.29
Noisy Training + Inference (0.4m)	31.06 ± 2.64	17.40 ± 0.69	17.56 ± 0.64	15.95 ± 0.70	17.45 ± 0.07

GeoMamba-WM exhibited little degradation under geometry-localization drift. Success only decreased by at most 3.34% for aligned and inconsistent scenarios, demonstrating how the SSM is able to account for this geometric drift in both trained and zero-shot scenarios. Training with odometry drift in the training data yields slight improvements in results at 0.1m, but does not have as significant of an effect at 0.4m. Ultimately, this ablation indicates that GeoMamba-WM can perform well with odometry noise and is not reliant on perfect samples of the geometric prior.

A.5.5 Deformability Label

The deformability label is included in the point cloud input to help the geometry encoder understand what surfaces may or may not be deformable. However, this value may not always be available in a pre-mapped environment. We therefore run GeoMamba-WM without deformability semantic labels during both variational-data finetuning and inference to evaluate whether the model is still effective.

Table 18: Comparison of GeoMamba-WM with and without the deformability label in its geometry input. Values report mean ± standard deviation over three evaluation seeds, in percentages.

Method	Aligned	Inconsistent	Geometry Marginal	Deformation Marginal	Noise Marginal
Success Rate (%)					
Ours	37.78 ± 6.94	**37.06 ± 1.71**	**33.75 ± 0.63**	**43.06 ± 1.20**	**36.34 ± 4.07**
No Deformability Label	**38.89 ± 5.09**	36.17 ± 0.92	32.50 ± 1.25	40.28 ± 2.17	34.26 ± 1.60
SoftSPL (%)					
Ours	**33.03 ± 2.20**	**18.84 ± 0.63**	**17.94 ± 0.48**	**17.25 ± 0.86**	**18.39 ± 0.48**
No Deformability Label	31.78 ± 2.68	18.58 ± 0.64	17.64 ± 0.28	16.15 ± 0.27	17.44 ± 0.36

Results indicate improved aligned planning performance and strong performance in inconsistent scenarios, with slight decreases across all metrics other than aligned success rate. The deformation marginal is most affected by removing the deformability label, showing the largest success rate decrease of 2.78%.

We measure the change in latents after removing the deformability label on 1200 held-out clips and report the results in table 19. CKA similarity between the two conditions remains extremely high, especially for the fused latent, and the effective rank of the geometry, corrected geometry, and fused

latents only decreases very slightly without the deformability label, explaining why a majority of the performance is retained without it.

Table 19: Effect of removing the deformability input on latent representations in deformable conditions.

Representation	CKA	Eff. Rank w/ Def.	Eff. Rank w/o Def.
Geometry, Z_g	0.9968	1.77	1.74
Corrected geometry, $\hat{Z}_g$	0.9968	1.79	1.76
Fused latent, Z_f	0.9997	35.35	35.31

A.6 AUXILIARY HEAD

A.6.1 ABLATION

GeoMamba-WM uses auxiliary heads in training time to predict the SSM's hidden state from the encoded and predicted latents. Specifically, it uses a single shared auxiliary head, in which the gradient from the encoded fused latent is used to update the weights of the head and the gradient from the predicted fused latent updates the predictive system. We ablate the use of the encoded latent auxiliary gradient to update the predictive system. The predictor and fusion objectives are, therefore, no longer identical.

$$\mathcal{L}_{P_\theta} = ||\hat{z}_{f,t+1} - z^{\text{target}}_{f,t+1}||_2^2 + \lambda_{A2}\mathcal{L}_{\text{Aux},t+1}, \qquad \mathcal{L}_{\text{Fusion}} = \mathcal{L}_{P_\theta} + \lambda_{A1}\mathcal{L}_{\text{Aux},t}$$

Table 20: Comparison of auxiliary head architectures on GeoMamba-WM in the GOI-NAV benchmark. Values report mean ± standard deviation over three evaluation seeds, in percentages.

Method	Aligned	Inconsistent	Geometry Marginal	Deformation Marginal	Noise Marginal
Success Rate (%)					
Ours	**37.78 ± 6.94**	**37.06 ± 1.71**	**33.75 ± 0.63**	**43.06 ± 1.20**	**36.34 ± 4.07**
Coupled Aux Heads	27.78 ± 6.94	22.70 ± 2.01	20.21 ± 3.08	32.29 ± 2.76	19.68 ± 2.81
No Aux Heads	23.33 ± 3.33	33.51 ± 2.77	29.38 ± 2.72	36.11 ± 2.41	31.48 ± 2.81
SoftSPL (%)					
Ours	**33.03 ± 2.20**	18.84 ± 0.63	17.94 ± 0.48	**17.25 ± 0.86**	18.39 ± 0.48
Coupled Aux Heads	15.53 ± 1.95	11.51 ± 0.75	11.15 ± 0.60	15.12 ± 0.78	10.59 ± 0.76
No Aux Heads	28.97 ± 0.34	**19.15 ± 1.02**	**18.39 ± 0.80**	15.44 ± 1.50	**18.55 ± 1.02**

Results indicate that the propagation of the encoded latents' gradient (Coupled Aux Heads) yields poorer results than GeoMamba-WM, or even the Base SSM model without any auxiliary heads. Compared to the model without auxiliary heads, our model does show slightly lowered SoftSPL in inconsistent scenarios, but reports increased success rates across all evaluations, led particularly by improved performance with deformable geometry. This motivates the use of the hidden state auxiliary head in GeoMamba-WM.

A.6.2 EFFECT ON LATENT SPACE

To better understand how the auxiliary heads structure the latent space to retain information captured by the SSM, we analyze the latent representations.

Table 21: Latent alignment and prediction quality across auxiliary head ablations. Effective rank measures the dimensional diversity of the predicted representation.

Metric	No Aux Heads	Coupled Aux	**Ours**
Pred. → Encoded CKA ↑	**0.854**	0.843	0.850
Pred. → Encoded MSE ↓	0.983	1.155	**0.967**
Predicted Effective Rank ↑	8.53	7.45	**9.36**
Encoded → EMA CKA ↑	0.991	0.980	**0.994**
Encoded → EMA MSE ↓	0.159	0.283	**0.094**

First, to diagnose the poorer performance of the coupled auxiliary head, we sample 2500 encoded, predicted, and EMA latents uniformly across all variations of the environment, and compute CKA similarity scores, effective rank, and MSE between the different latents, as seen in table 21. Two areas of representation drift are apparent with the coupled aux. First, the predicted and encoded latent spaces drift apart. Second, the encoded latent space is more dissimilar to the EMA target used as the model objective. The combination of these two drifts results in partial representation collapse, whereby the coupled aux model has the lowest effective rank for the predicted latents, which yields the lowest success rates of all the models. Using the same objective for the entire system, whether it

be with or without the auxiliary supervision, gives better results by preventing this encoder-predictor representation drift. As shown, our model has improved latent space alignment between the encoded, predicted, and EMA latent spaces, and the highest effective rank for the predicted latents, indicating the richest representation of all the ablations.

Another way to understand this retention is to look at correlations for linear probes on the fused latents of models with and without the hidden state. In table 22, we can see the performance of the probes on the encoded latent, as well as autoregressively predicted latents for various timesteps. While initial encoded latents hold similar spatial information across models, the addition of the auxiliary objective significantly improves the retention of geometric features across longer prediction horizons. This translates into better planning performance for models with auxiliary objectives. As an aside, we can observe that the SSM also improves the predicted latent quality over the model without it.

Table 22: Mean linear probe R^2 for 9 spatial coordinate targets, averaged over 10 environment conditions.

Latent State	No SSM	Base SSM	**Hidden-aux SSM**
Encoded, $h = 0$	0.750	**0.763**	<u>0.762</u>
Predicted, $h = 1$	0.673	**0.690**	<u>0.688</u>
Predicted, $h = 2$	0.579	<u>0.594</u>	**0.602**
Predicted, $h = 4$	0.345	<u>0.349</u>	**0.388**
Predicted, $h = 8$	0.026	<u>0.051</u>	**0.069**

A.7 CEM TARGET ABLATION

For planning, the default CEM target used is identical to DINO-WM. During rollouts, the predictor is used to autoregressively predict fused latents, bypassing the encoders, SSM, and fusion altogether. However, given that the auxiliary head supervision target is the hidden state, we can use this head during inference to predict the hidden state from these predicted latents, and either use that directly as the CEM target, or as an addition to the fused target. Furthermore, even in the Base SSM model trained without the auxiliary head, we can train just the auxiliary head on the frozen model and use it during planning to augment the target.

$$C_{\text{Hidden}} = ||\hat{H}_H - H_{\text{goal}}||_2^2, \qquad C_{\text{Fused+Hidden}}(\hat{z}_{f,H}) = ||\hat{z}_{f,H} - z_{f,\text{goal}}||_2^2 + C_{\text{Hidden}}$$

Table 23: Effect of the CEM ranking target on GOI-NAV performance. Values are percentages reported as three-seed mean ± standard deviation.

CEM Target	Aligned	Inconsistent	Geometry Marginal	Deformation Marginal	Noise Marginal
Success Rate (%)					
Fused Only (Ours)	37.78 ± 6.94	**37.06 ± 1.71**	**33.75 ± 0.63**	**43.06 ± 1.20**	**36.34 ± 4.07**
Fused + Hidden	**40.00 ± 6.67**	16.84 ± 3.20	16.67 ± 2.19	16.32 ± 2.62	17.82 ± 2.44
Hidden Only	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
SoftSPL (%)					
Fused Only (Ours)	**33.03 ± 2.20**	**18.84 ± 0.63**	**17.94 ± 0.48**	**17.25 ± 0.86**	**18.39 ± 0.48**
Fused + Hidden	28.17 ± 2.69	9.35 ± 0.67	9.61 ± 0.25	6.92 ± 0.68	9.32 ± 0.87
Hidden Only	1.18 ± 0.42	0.58 ± 0.15	0.64 ± 0.14	0.52 ± 0.20	0.58 ± 0.13

Results indicate that while adding the hidden state to the target does not have a significant effect in aligned scenarios, performance degrades in inconsistent scenarios. Using only the hidden target does not yield any meaningful success, indicating it does not have sufficient information for planning.

To analyze why, we use the three CEM targets to score 150 latents from GeoMamba-WM. The metrics in table 24 indicate that the fused latent is nearly universally superior. It has highest correlation with the oracle path, the selected elites have the lowest resultant distance to the goal, lowest regret relative to the oracle, and has the greatest overlap with the oracle. The hidden-only metric has negative, near-zero correlation with the oracle, indicating it is an exceptionally poor planning target, and is a confounding signal when added to the target along with the fused latent.

Table 24: Ranking quality comparisons of various CEM targets. All objectives score the same candidate sequences from the first CEM iteration.

Ranking Metric	**Fused**	Fused+Hidden	Hidden-Only
Spearman with Physical Oracle ↑	**0.299**	0.241	−0.053
Elite Physical Remaining Distance ↓	**1.976 m**	1.995 m	2.081 m
Elite Regret ↓	**0.215 m**	0.233 m	0.319 m
Oracle-Elite Overlap ↑	**9.83%**	9.54%	6.67%
Elite Goal Rate ↑	4.24%	**4.29%**	2.27%

A.8 Effect of History on SSM Performance

The SSM hidden state has two distinct effects in GeoMamba-WM: first, maintaining a consistent history state of previous actions, and second, providing a mechanism for observation-conditioned geometry correction to reduce inconsistency. To isolate the advantages of each of these, we run a planning-time ablation by resetting the hidden state at every environment step. This variant retains the geometry-correction mechanism, isolating its impact against the No-SSM baseline, and removes the recurrent history, revealing how retaining this hidden state improves planning in our model.

Table 25: Comparison of SSM history on GeoMamba-WM performance in the GOI-NAV benchmark. Values report mean ± standard deviation over three evaluation seeds, in percentages.

Method	Aligned	Inconsistent	Geometry Marginal	Deformation Marginal	Noise Marginal
Success Rate (%)					
Ours	**37.78 ± 6.94**	**37.06 ± 1.71**	**33.75 ± 0.63**	**43.06 ± 1.20**	**36.34 ± 4.07**
Reset-State SSM	25.56 ± 9.62	32.80 ± 2.73	28.96 ± 2.95	35.77 ± 5.35	31.71 ± 2.23
No SSM	33.33 ± 3.33	26.95 ± 2.40	25.42 ± 3.21	28.12 ± 3.12	23.38 ± 2.00
SoftSPL (%)					
Ours	**33.03 ± 2.20**	**18.84 ± 0.63**	**17.94 ± 0.48**	**17.25 ± 0.86**	**18.39 ± 0.48**
Reset-State SSM	29.69 ± 3.47	17.45 ± 0.72	16.57 ± 0.82	15.21 ± 1.08	17.11 ± 0.64
No SSM	29.14 ± 1.66	17.32 ± 0.91	17.23 ± 0.97	14.50 ± 1.32	16.08 ± 0.94

Results show that retaining a consistent history improves performance across all metrics, most significantly so in aligned scenes. The geometry correction alone yields a larger performance gain in inconsistent scenes, improving the No-SSM model across nearly all inconsistent metrics. In aligned scenes, the correction reduces success rate, which is recovered by the maintenance of a recurrent history.

To better determine where this improvement originates, we additionally compare the internal representations produced by the two runs on 2500 held-out clips balanced across all conditions. The CKA similarity between the two hidden states decreases over time, and the rank of the persistent hidden state increases over time while the reset-state rank does not. This indicates greater representation complexity given a recurrent hidden state that cannot be captured by instantaneous geometry and observations.

Table 26: Representation similarity and effective rank of persistent and reset SSM states on held-out clips. MSE measures the difference between the two hidden states.

Timestep	Hidden CKA	Hidden MSE	Persistent rank	Reset rank
$t = 0$	1.000	0.000	1.28	1.28
$t = 1$	0.517	0.604	1.72	1.51
$t = 2$	0.301	0.948	2.17	1.45
$t = 3$	0.206	1.218	2.41	1.35

Interestingly, we find that this difference in the hidden state does not propagate to large changes in the corrected geometry or fused latent. Table 27 indicates that they are nearly identical in both cases. However, the minimal differences that are present in the fused latent affect action selection during CEM, which accumulates throughout the rollout to yield improved planning success.

Table 27: Effect of SSM history after the hidden state is projected into corrected geometry and fused latent spaces.

Diagnostic	Persistent vs. Reset CKA	Persistent vs. Reset MSE
Hidden state, $t = 3$	0.206	1.218
Corrected geometry, $t = 3$	1.000	6.61×10^{-4}
Fused latent, $t = 3$	0.9997	1.24×10^{-3}

A.9 World Model Target Ablation

GeoMamba-WM uses a static fused target for the model objective, produced from an observation latent and uncorrected geometry latent. However, the model uses an SSM to produce corrected geometry latents from the input priors, which are fused with the observations to create the input fused latents. This creates an asymmetry between the input and target branches, which does not dynamically correct for geometric inconsistencies. We therefore ablate the world model objective by evaluating a dynamic, stateful target. In this variant, an EMA copy of the SSM is introduced, which maintains its own recurrent hidden state and refines the geometry before the EMA fusion module creates the fused latent target:

$$\begin{aligned} \bar{H}_t, \hat{Z}_{g,t}^{\mathrm{EMA}} &= \mathrm{SSM}_{\mathrm{EMA}}\left(\bar{H}_{t-1}, Z_{g,t}, Z_{o,t}, a_{t-1}\right) \\ z_{f,t}^{\mathrm{target}} &= \mathrm{MeanPool}\left(\mathrm{sg}\left(\mathrm{Fusion}_{\mathrm{EMA}}\left(Z_{o,t}, \hat{Z}_{g,t}^{\mathrm{EMA}}\right)\right)\right) \end{aligned}$$

Note that this ablation does not receive any additional privileged geometric or dynamic information than GeoMamba-WM.

Table 28: Comparison of static and dynamic world model targets. Values report mean ± standard deviation over three evaluation seeds, in percentages.

Method	Aligned	Inconsistent	Geometry Marginal	Deformation Marginal	Noise Marginal
Success Rate (%)					
Uncorrected Target (Ours)	37.78 ± 6.94	**37.06 ± 1.71**	**33.75 ± 0.63**	**43.06 ± 1.20**	**36.34 ± 4.07**
Corrected Target	**40.00 ± 5.77**	32.80 ± 2.68	28.54 ± 2.53	38.89 ± 3.35	32.64 ± 2.51
SoftSPL (%)					
Uncorrected Target (Ours)	**33.03 ± 2.20**	**18.84 ± 0.63**	**17.94 ± 0.48**	**17.25 ± 0.86**	**18.39 ± 0.48**
Corrected Target	31.67 ± 3.73	18.17 ± 1.30	17.24 ± 1.09	16.49 ± 1.51	17.69 ± 0.96

This corrected target slightly increases the aligned success rate from 37.78% to 40.00%, but reduces the success rates in every inconsistent evaluation. It also reduces SoftSPL across all categories.

These results suggest that applying a recurrent geometric correction to both the prediction and the target weakens the separation between a stable geometric reference and the SSM-based correction learned by the model. Additionally, the dynamic target may produce a more unstable objective for the system, which altogether results in poorer planning. This motivates the use of a static target for GeoMamba-WM.

A.10 Curriculum Training Ablation

GeoMamba-WM is trained with a very specific curriculum training procedure to yield optimal results. We ablate two different variational training schemes. First, we train the model directly on variational data from scratch. Results indicate that inconsistencies in this data prevent the model from properly learning associations between geometry and observations, resulting in poor results across the board.

Second, we implement a curriculum training method from scratch, where the model is trained on fully aligned data for 5 epochs, a 90/10 ratio of aligned to variational data for the next 5 epochs, followed by 80/20 and 70/30 for 5 more epochs each. This curriculum yields superior results to the direct variational training, and even outperforms our model in aligned success rate and SoftSPL in deformable environments. Regardless, our 2-stage curriculum training protocol provides the most consistent results across most metrics, motivating its selection as the final training method.

Table 29: Comparison of training curricula for GeoMamba-WM on the GOI-NAV benchmark. Values report mean ± standard deviation over three evaluation seeds, in percentages.

Method	Aligned	Inconsistent	Geometry Marginal	Deformation Marginal	Noise Marginal
Success Rate (%)					
Aligned Training Only	**55.56 ± 11.71**	27.31 ± 0.61	25.00 ± 0.00	32.29 ± 2.08	25.00 ± 0.69
Direct Variational Training	7.78 ± 5.09	13.12 ± 5.35	11.46 ± 5.32	21.53 ± 10.12	8.33 ± 4.86
From-Scratch Curriculum	38.89 ± 10.18	27.66 ± 4.54	25.42 ± 5.64	39.24 ± 6.78	25.93 ± 4.19
Ours	37.78 ± 6.94	**37.06 ± 1.71**	**33.75 ± 0.63**	**43.06 ± 1.20**	**36.34 ± 4.07**
SoftSPL (%)					
Aligned Training Only	**36.79 ± 4.82**	14.71 ± 0.49	14.18 ± 0.29	13.43 ± 0.31	13.47 ± 0.76
Direct Variational Training	8.18 ± 3.02	10.01 ± 0.79	9.45 ± 0.51	12.55 ± 1.92	7.60 ± 1.08
From-Scratch Curriculum	22.41 ± 3.12	14.58 ± 1.20	13.72 ± 1.41	**18.35 ± 1.47**	14.30 ± 1.34
Ours	33.03 ± 2.20	**18.84 ± 0.63**	**17.94 ± 0.48**	17.25 ± 0.86	**18.39 ± 0.48**

A.11 SSM Ablation Results

Expanding on SSM ablations in the main text, we also report the standard deviation for each measurement.

Table 30: Performance comparisons between various geometry encoders and SSM implementations, with standard deviations reported across 3 CEM seeds.

Method	Aligned	Inconsistent	Geometry Marginal	Deformation Marginal	Noise Marginal
Success Rate (%)					
No SSM	33.33 ± 3.33	26.95 ± 2.40	25.42 ± 3.21	28.12 ± 3.12	23.38 ± 2.00
SSM, No Aux Head	23.33 ± 3.33	33.51 ± 2.77	29.38 ± 2.72	36.11 ± 2.41	31.48 ± 2.81
Observation-Only SSM, No Aux Head	6.67 ± 6.67	5.67 ± 2.68	5.21 ± 2.60	2.43 ± 1.59	6.48 ± 3.13
DINO-WM	4.44 ± 3.85	2.48 ± 1.11	2.29 ± 1.30	2.43 ± 1.59	2.78 ± 0.70
Ours (SSM, Aux Head)	**37.78 ± 6.94**	**37.06 ± 1.71**	**33.75 ± 0.63**	**43.06 ± 1.20**	**36.34 ± 4.07**
SoftSPL (%)					
No SSM	29.14 ± 1.66	17.32 ± 0.91	17.23 ± 0.97	14.50 ± 1.32	16.08 ± 0.94
SSM, No Aux Head	28.97 ± 0.34	**19.15 ± 1.02**	**18.39 ± 0.80**	15.44 ± 1.50	**18.55 ± 1.02**
Observation-Only SSM, No Aux Head	19.48 ± 2.01	10.25 ± 0.61	10.08 ± 0.62	4.46 ± 0.35	9.79 ± 0.55
DINO-WM	9.00 ± 2.20	6.14 ± 0.55	5.98 ± 0.61	6.28 ± 0.54	5.31 ± 0.47
Ours (SSM, Aux Head)	**33.03 ± 2.20**	18.84 ± 0.63	17.94 ± 0.48	**17.25 ± 0.86**	18.39 ± 0.48