G^{2}WAM: World Action Models for Efficient and Geometry Grounded Image-Goal Navigation

Anonymous authors
Paper under double-blind review

Abstract

World models capture environment dynamics and support counterfactual reasoning for planning. In image-goal navigation, where the target often lies beyond the current field of view, the agent must not only model environmental dynamics but also maintain a persistent spatial memory of the scene. Existing approaches either introduce spatial context by fine-tuning pretrained models on the target task, which underexploits their spatial priors, or rely on external 3D reconstruction, which is computationally expensive for real-world deployment. In this paper, we propose the Geometry Grounded World Action Model (G^{2}WAM) for image-goal navigation. G^{2}WAM encodes spatial cues through a learned scene-coordinate localization module that anchors the current, goal, and history frames in a common reference frame, yielding a stable and consistent spatial context that enables efficient navigation without explicit 3D reconstruction. Following the world-action-model paradigm, G^{2}WAM jointly predicts future states and actions, eliminating the need for costly cross-entropy method (CEM) style planning. In addition, G^{2}WAM operates entirely in latent space, avoiding the computational burden of pixel-level reconstruction and substantially reducing inference cost. Across four trajectory prediction benchmarks, G^{2}WAM reduces average ATE and RPE by 40.1% and 35.2%, respectively. In closed-loop navigation experiments, it improves the success rate from 54.0% to 63.0% and the success weighted by path length (SPL) from 36.4% to 59.2%, while maintaining high efficiency.

1 Introduction

Image-goal navigation asks an agent to reach the location shown in a target image by using its stream of first-person observations. Recent advances have progressed from reactive end-to-end policies (Shah et al., 2023a; Sridhar et al., 2024; Shah et al., 2023b) to world-model-based approaches (Bar et al., 2025; Zhou et al., 2025; Zhang et al., 2025b; Hafner et al., 2025), in which the world model serves as a learned model of environment dynamics. Concretely, it can jointly capture navigation priors and scene memory within a single training pipeline, and support counterfactual reasoning that helps the agent solve complex long-horizon navigation tasks, including those whose targets lie beyond the current field of view. In practice, however, error accumulation over multi-step rollouts causes long-horizon imagination to degrade rapidly (Bar et al., 2025; Luan et al., 2026). Moreover, cross-entropy method (CEM) planning requires repeated forward evaluations, incurring high inference cost and latency, and the selected open-loop trajectory must still be converted into closed-loop waypoints at execution time, which introduces additional error.

Existing methods attempt to explicitly inject depth information during training (Wu et al., 2026), or to bolt on external 3D reconstruction (Ren et al., 2025; Wang et al., 2026a), Gaussian-splatting (Ma et al., 2026), or latent spatial memory (Wu et al., 2025; Oshima et al., 2026; Wang et al., 2026b), so as to provide the world model with explicit spatial context that improves long-horizon and spatial consistency. Yet these methods either exploit spatial priors insufficiently or are too heavy for practical deployment. Our key insight is that, when navigating, humans do not maintain a precise geometric model of the environment (Warren, 2019; Ishikawa & Montello, 2006); instead, they rely on an egocentric, reactive representation (Wang & Spelke, 2002), which can be captured by the relative pose between the current view and the goal image.

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.

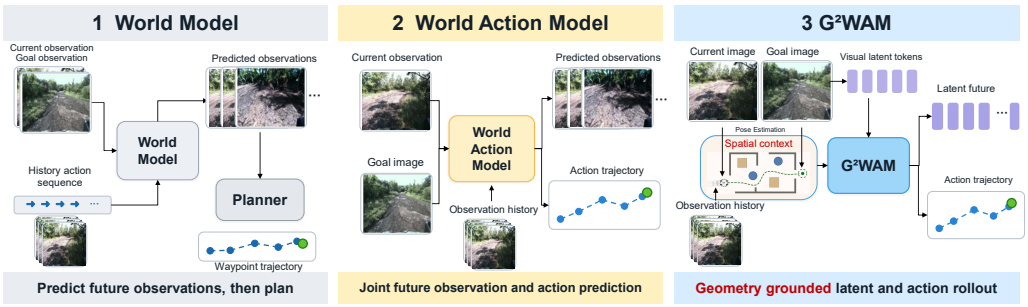


Figure 1: Navigation world model paradigms. Conventional world models plan over predicted observations, whereas world action models jointly predict observations and actions. G^{2}WAM further leverages estimated spatial context to efficiently roll out future latents and executable actions.

We therefore present the Geometry Grounded World Action Model (G^{2}WAM) for image-goal navigation. G^{2}WAM encodes environmental information into an end-to-end localization network and provides the agent with egocentric relative pose to the goal as well as the history trajectory as spatial cues. By anchoring all observation frames and the goal frame to fixed positions in space, we obtain a stable, geometrically consistent spatial context. In designing G^{2}WAM, we paid particular attention to its efficiency for practical deployment. To this end, G^{2}WAM adopts the world-action-model paradigm to jointly predict action chunks and latent future states, sidestepping the expensive CEM-style planning. It further keeps the entire rollout in latent space, avoiding pixel-level future reconstruction. Fig. 1 depicts the key difference between our method and existing methods based on world models and world action models.

Our contributions can be summarized as follows:

- We propose G^{2}WAM, a novel implicit world action model for navigation that jointly predicts latent future states and executable action chunks, avoiding the costly CEM-style planning and heavy pixel-level reconstruction.
- We introduce a new paradigm for spatial context aiding. Instead of reconstructing explicit geometry, we anchor the observation and goal frames in a shared end-to-end localization network, which efficiently provides a continuous, geometrically consistent spatial context over the navigation history and the relative target.
- Extensive experiments demonstrate that G^{2}WAM achieves the best overall trajectory prediction and navigation performance, reducing average ATE and RPE by 40.1% and 35.2% in trajectory prediction, and improving the success rate from 54% to 63% (a 9-point gain) and success weighted by path length (SPL) from 36.4% to 59.2% (a 22.8-point gain) in closed-loop navigation, while maintaining high efficiency. We will open-source our approach to benefit the research community.

2 Related Work

2.1 Visual Navigation

Experience-based navigation uses recorded routes, demonstrations, or topological graphs to guide motion (Furgale & Barfoot, 2010; Pathak et al., 2018; Savinov et al., 2018; Shah et al., 2021a). Goal-conditioned policies instead map current and goal images directly to actions; representative methods include GNM (Shah et al., 2023a), ViNT (Shah et al., 2023b), and NoMaD (Sridhar et al., 2024). These methods do not explicitly predict future world states. Metric goal estimation provides complementary geometric guidance. PointGoal methods use localization or visual odometry (Wijmans et al., 2020; Partsey et al., 2022), while image-based localization estimates pose through feature matching or scene-coordinate regression (Sarlin et al., 2020; Brachmann et al., 2017; Jiang et al., 2025). AnyImageNav (Deng et al., 2026) uses recovered goal pose for precise navigation. Reactive policies have achieved promising results, but they lack modeling of environment dynamics, mak-

ing them prone to pattern matching. Accordingly, recent state-of-the-art methods have progressed toward world-model-based approaches.

2.2 World Models and World Action Models for Navigation

Navigation world models predict the consequences of actions in image or latent space. NWM (Bar et al., 2025) generates future images, whereas DINO-WM (Zhou et al., 2025), LS-NWM (Zhang et al., 2025a), and ReL-NWM (Zhang et al., 2025b) support planning with predicted visual features or latent states. World action models further couple state and action prediction. Spatial-perceiving WAM (Chen et al., 2026) jointly predicts RGB-D observations and waypoints, NavWAM (Azuma et al., 2026) models observations, goal progress, and action chunks, and WAMNav (Yang et al., 2026) jointly predicts latent dynamics and actions. G^{2}WAM also performs joint latent state-action prediction, but introduces explicit spatial context through relative pose and causal motion conditioning. Unlike existing WAMs, it predicts metric waypoints through an implicit latent rollout without reconstructing future observations.

Geometry-aware world models mainly improve spatial consistency in two ways. The first direction maintains geometry-grounded spatial memory. GEN3C (Ren et al., 2025) and EvoWorld (Wang et al., 2026a) construct explicit 3D scene presentations and reproject them into target views to condition visual generation. Other methods improve long-term consistency by organizing historical information according to scene geometry: long-term spatial memory (Wu et al., 2025) retrieves previously observed scene content, WorldPack (Oshima et al., 2026) selects and compresses observations using camera pose and field-of-view overlap, and Mirage (Wang et al., 2026b) stores depth-unprojected latent tokens in a persistent 3D cache. The second direction introduces geometric constraints without maintaining a persistent scene representation. Geometry Forcing (Wu et al., 2026) aligns intermediate generative features with geometric representations during training, while DR-NWM (Luan et al., 2026) uses future anchors and bidirectional epipolar guidance to reduce geometric drift during visual rollout.

In contrast, G^{2}WAM neither reconstructs an explicit 3D scene nor generates future images under geometric constraints. Instead, it derives spatial context from estimated poses, a lightweight metric condition for jointly rolling out latent states and executable actions.

3 Method

We build G^{2}WAM around a single design principle: efficient, geometry-grounded navigation. This principle is realized through two design choices. First, the spatial context is geometry-grounded without costly 3D reconstruction. A lightweight scene-coordinate localization module anchors the goal and observed frames in a shared scene frame, so the relative pose between the current view and the goal, together with the history motions, provides an egocentric geometric context that grounds the rollout in the target. Second, the model is efficient by construction. It adopts the world-action-model paradigm and rolls out entirely in latent space, avoiding CEM-style planning and pixel-level reconstruction. Together, an implicit world action model consumes this lightweight relative-geometry context and rolls it forward, jointly predicting latent future states and canonical waypoint increments. The overall architecture is shown in Fig. 2.

We first describe the spatial context induced by scene-coordinate anchoring (Sec. 3.1), then the implicit world action model (Sec. 3.2), and finally how relative-pose and motion conditioning adapt the model to a concrete platform (Sec. 3.3). Training is described in Sec. 3.4.

3.1 Spatial Context via Scene-Coordinate Anchoring

Our key insight is that an agent need not build an explicit geometric model of the environment to navigate. It needs a stable, geometrically consistent spatial context indicating where the goal lies relative to its history of visited locations. We obtain this by anchoring the observation history and the goal image in a shared scene coordinate through an end-to-end localization network.

Concretely, the localization module estimates the current, goal, and history-frame poses in a shared scene frame, from which we compose the current-to-goal relative transformation and the consecutive motion increments. Let $\boldsymbol{I}_j$ be the image at index j, and let $\hat{\boldsymbol{T}}_j^{\mathrm{S}} \in \mathrm{SE}(3)$ map its calibrated robot

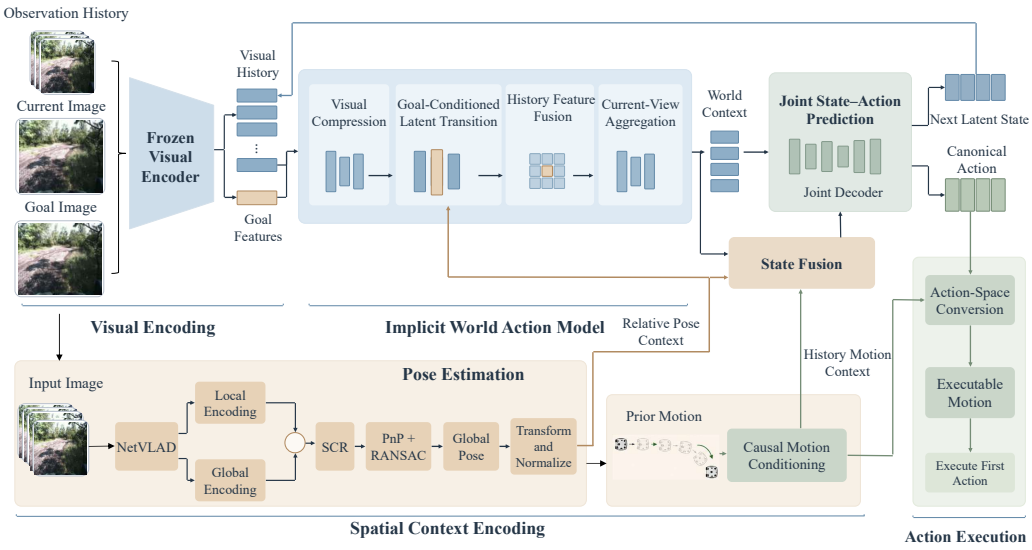


Figure 2: Overview of G^{2}WAM. A learned localizer anchors the goal and observed frames in a shared scene frame to obtain the goal-relative pose and causal history motions. The goal image and pose guide joint latent and waypoint prediction, while the history motions condition the action branch.

frame into the shared scene frame S, where t, g, and i index the current, goal, and history frames. We first obtain each pose through a scene-specific image localization module f_{loc}:

$$\hat{\boldsymbol{T}}_j^{\mathrm{S}} = f_{\mathrm{loc}}(\boldsymbol{I}_j; \mathcal{D}_{\mathrm{S}}), \qquad j \in \{t-3, \ldots, t, g\}, \tag{1}$$

where $\mathcal{D}_{\mathrm{S}}$ denotes pre-collected scene data and camera-to-robot calibration is included in the interface. We then compute

$$\hat{\boldsymbol{T}}_{t\to g} = (\hat{\boldsymbol{T}}_t^{\mathrm{S}})^{-1}\hat{\boldsymbol{T}}_g^{\mathrm{S}}, \qquad \hat{\boldsymbol{T}}_{i\to i+1} = (\hat{\boldsymbol{T}}_i^{\mathrm{S}})^{-1}\hat{\boldsymbol{T}}_{i+1}^{\mathrm{S}}, \quad i = t-3, \ldots, t-1. \tag{2}$$

The planar translation of $\hat{\boldsymbol{T}}_{t\to g}$ is converted to the model's coordinate convention and divided by the waypoint spacing ℓ to form $\boldsymbol{p}_t$; the consecutive transforms give the history-dependent entries of $\boldsymbol{m}_t$. These are embedded as $\boldsymbol{e}_t^{\mathrm{pose}}$ and $\boldsymbol{e}_t^{\mathrm{motion}}$ (Sec. 3.3) and condition the latent rollout in Sec. 3.2. This design separates the world action model from the localization network: since the model consumes only the current-to-goal relative transformation and recent causal motion, the localization network can be any end-to-end method or a traditional place-recognition-plus-pose-estimation pipeline, without changing the architecture. This plug-and-play property makes the world action model agnostic to the localization interface. By default, we use R-SCoRe (Jiang et al., 2025), a state-of-the-art image-based end-to-end localizer that maps images to geometrically grounded correspondences for robust 6-DoF pose estimation, achieving 64.3% success within $(0.25\,\mathrm{m}, 2^\circ)$ at 140–270 ms per query on an RTX 2080.

Note that scene-coordinate anchoring assumes a previously seen scene, as it requires pre-collected trajectory data. This is practically meaningful because our method targets out-of-field-of-view navigation. In a completely unknown scene, such a task can rely only on a world model's imagination, which is empirical and unreliable since any configuration is possible. Existing methods therefore resort to fine-tuning or external 3D reconstruction. Our method instead anchors observations and the goal to fixed spatial positions via end-to-end localization, exploiting scene information more directly than fine-tuning and more lightly than 3D reconstruction.

3.2 Implicit World Action Model

The world action model is built to roll out task-relevant state without ever reconstructing an image. This choice is deliberate: navigation only requires the model to track where the agent is heading, not to reproduce the appearance of the scene, so an abstract latent rollout suffices. We therefore predict

the next patch latent and the corresponding action from a shared goal-conditioned transition context, never passing through an image decoder. The relative pose $\boldsymbol{p}_t$ and motion profile $\boldsymbol{m}_t$ from Sec. 3.1 enter this model through their embeddings $\boldsymbol{e}_t^{\text{pose}}$ and $\boldsymbol{e}_t^{\text{motion}}$, defined in Sec. 3.3.

A frozen DINOv3 encoder (Siméoni et al., 2025) maps the observation history and goal image into patch latents $\boldsymbol{z}_{t-H+1:t} \in \mathbb{R}^{H\times P\times D}$ and $\boldsymbol{z}_g \in \mathbb{R}^{P\times D}$, where $H = 4$ is the context length, P the number of patches, and $D = 384$ the latent dimension. The goal embedding is $\boldsymbol{q}_g = \phi_g(\text{pool}(\text{LN}(\boldsymbol{z}_g)))$, where ϕ_g is a two-layer MLP, LN is layer normalization, and pool denotes mean pooling over patches. At rollout step $k \in \{0, \dots, 7\}$, the model maintains a fixed-length latent history and computes

$$\begin{aligned} \boldsymbol{c}_{t+k} &= \text{Transition}(\boldsymbol{z}_{t+k-H+1:t+k}, \boldsymbol{q}_g, \boldsymbol{e}_t^{\text{pose}}), \\ \boldsymbol{s}_{t+k} &= \text{pool}(\boldsymbol{c}_{t+k}) + \boldsymbol{e}_t^{\text{pose}} + \boldsymbol{e}_t^{\text{motion}}, \\ \big(\boldsymbol{z}_{t+k+1}^{\mathcal{J}}, \hat{\boldsymbol{a}}_{t+k}^{\text{can}}\big) &= \mathcal{J}(\boldsymbol{c}_{t+k}, \boldsymbol{s}_{t+k}), \\ \hat{\boldsymbol{z}}_{t+k+1} &= \text{LN}\big(\alpha_{\mathcal{J}} \boldsymbol{z}_{t+k+1}^{\mathcal{J}} + \alpha_{\text{B}} \phi_{\text{B}}(\boldsymbol{c}_{t+k})\big). \end{aligned} \tag{3}$$

The transition module fuses the latent history with the goal and relative-pose conditions to produce the patch-level context $\boldsymbol{c}_{t+k} \in \mathbb{R}^{P\times D}$. Its pooled representation is combined with the pose and motion embeddings to obtain the compact step state $\boldsymbol{s}_{t+k} \in \mathbb{R}^D$. The joint decoder $\mathcal{J}$ then predicts the next latent candidate and the canonical waypoint increment $\hat{\boldsymbol{a}}_{t+k}^{\text{can}} = (\Delta x, \Delta y, \Delta\theta)$. As defined in Eq. 3, the latent candidate is combined with a base transition and appended to the fixed-length history, yielding an autoregressive latent rollout without image decoding. Detailed transition, conditioning, and decoder architectures are provided in Appendix A.

3.3 Spatial Context Conditioning And Action Adaption

The two conditioning signals play distinct roles: the relative pose supplies target geometry, while the motion profile records the past trajectory and specifies how the canonical prediction is converted into executable motion.

The relative-pose conditioner receives $\boldsymbol{p}_t = (\Delta x_t^g, \Delta y_t^g, \sin\Delta\theta_t^g, \cos\Delta\theta_t^g)$. This four-dimensional vector is derived from $\hat{\boldsymbol{T}}_{t\to g}$ in Eq. 2. Here Δx_t^g and Δy_t^g are the current-to-goal planar translation in the model's coordinate convention and in waypoint units, while $\Delta\theta_t^g$ is the relative yaw. The conditioner separates translation from rotation and computes

$$\boldsymbol{e}_t^{\text{pose}} = \phi_{\text{fuse}}([\phi_{\text{trans}}(\boldsymbol{u}_t, \log(1+d_t)), \phi_{\text{yaw}}(\boldsymbol{v}_t)]), \tag{4}$$

where $\boldsymbol{u}_t = (\Delta x_t^g, \Delta y_t^g)/\max(d_t, \epsilon)$ is the normalized goal-direction vector, $d_t = \sqrt{(\Delta x_t^g)^2 + (\Delta y_t^g)^2}$ is the goal distance in waypoint units, $\boldsymbol{v}_t = (\sin\Delta\theta_t^g, \cos\Delta\theta_t^g)$ is the relative-yaw vector, and $\epsilon > 0$ prevents division by zero. The brackets denote concatenation. The functions ϕ_{trans} and ϕ_{yaw} are separate two-layer MLPs. The fusion function ϕ_{fuse} applies a two-layer MLP followed by layer normalization.

Independently, the motion conditioner receives the five-dimensional profile formed from the consecutive transforms in Eq. 2, $\boldsymbol{m}_t = [\bar{d}_t, \bar{\theta}_t, d_s, \theta_s, \ell]^{\mathsf{T}}$. The history-dependent terms are the median per-step distance $\bar{d}_t$, divided by waypoint spacing ℓ, and the mean absolute per-step yaw change $\bar{\theta}_t$. The dataset-level translation and yaw scales are d_s and θ_s, respectively; ℓ is the waypoint spacing in metres. A two-layer MLP with layer normalization maps this profile to the motion embedding $\boldsymbol{e}_t^{\text{motion}} = \phi_{\text{motion}}(\log(\mathbf{1} + \boldsymbol{m}_t))$, where the logarithm is applied componentwise.

A canonical-motion adapter A maps each canonical prediction to the action interface:

$$\hat{\boldsymbol{a}}_{t+k} = A\big(\hat{\boldsymbol{a}}_{t+k}^{\text{can}}, \boldsymbol{m}_t\big) = \Big[\mathcal{N}_{xy}\Big(\tilde{d}_t \hat{\boldsymbol{a}}_{t+k,xy}^{\text{can}}\Big); \tilde{\theta}_s \hat{a}_{\theta,t+k}^{\text{can}}\Big]. \tag{5}$$

Here $\hat{\boldsymbol{a}}_{t+k,xy}^{\text{can}}$ denotes the two-dimensional canonical translation, $\hat{a}_{\theta,t+k}^{\text{can}}$ the scalar canonical yaw, $\tilde{d}_t = \text{clip}(\bar{d}_t, 0.1, 4.0)$, and $\tilde{\theta}_s = \text{clip}(\theta_s, 0.05, 1.0)$. The semicolon denotes concatenation. For a two-dimensional input $\boldsymbol{\xi}$, the planar normalization is $\mathcal{N}_{xy}(\boldsymbol{\xi}) = 2(\boldsymbol{\xi} - \boldsymbol{b}_{\min}) \oslash (\boldsymbol{b}_{\max} - \boldsymbol{b}_{\min}) - \mathbf{1}$, where $\boldsymbol{b}_{\min}$ and $\boldsymbol{b}_{\max}$ are fixed two-dimensional action bounds, $\oslash$ denotes componentwise division, and $\mathbf{1}$ is the two-dimensional all-ones vector. This function maps translation in waypoint units to the normalized planar action interface. Thus $\boldsymbol{p}_t$ supplies target geometry, whereas $\boldsymbol{m}_t$ specifies how the canonical prediction is converted into platform-specific motion.

3.4 TRAINING

We train the G^{2}WAM end-to-end with a multi-term objective, while the localization interface is trained separately. The latent loss combines a cosine term and a small mean-squared-error term, the action loss is a smooth-ℓ_1 on the canonical action, and trajectory supervision compares cumulative waypoint coordinates and yaw over the rollout. Rare turns receive extra weight so the model does not collapse to always-advance behavior:

$$\mathcal{L} = \lambda_z \mathcal{L}_z + \lambda_a \mathcal{L}_a + \lambda_{\text{traj}} \mathcal{L}_{\text{traj}} + \lambda_{\text{yaw}} \mathcal{L}_{\text{yaw}}, \tag{6}$$

The two direct prediction terms are

$$\begin{aligned} \mathcal{L}_z &= 1 - \text{Cos}(\hat{\boldsymbol{Z}}, \boldsymbol{Z}^*) + 0.3\,\text{MSE}(\hat{\boldsymbol{Z}}, \boldsymbol{Z}^*), \\ \mathcal{L}_a &= \frac{\sum_{b,k} w_{b,k}\, \ell_{\text{SL1}}(\hat{\boldsymbol{a}}^{\text{can}}_{b,t+k}, \boldsymbol{a}^{\text{can},*}_{b,t+k})}{\sum_{b,k} w_{b,k}}, \end{aligned} \tag{7}$$

where $\hat{\boldsymbol{Z}}$ and $\boldsymbol{Z}^*$ stack the predicted and target patch latents across rollout steps, Cos averages patchwise cosine similarity, MSE averages squared feature error, and ℓ_{SL1} averages smooth-ℓ_1 error over the three canonical-action components. The weight $w_{b,k}$ increases for turning steps according to the target executable yaw increment and applies to the entire action increment. The remaining terms $\mathcal{L}_{\text{traj}}$ and $\mathcal{L}_{\text{yaw}}$ compare cumulative executable waypoint positions and yaw using ℓ_1 losses, with planar increments unnormalized before accumulation. The scalar coefficients $\lambda_z, \lambda_a, \lambda_{\text{traj}}, \lambda_{\text{yaw}}$ weight these four losses, whose values are 0.3, 3.0, 2.0, and 0.5, respectively. We use AdamW with batch size 32, zero weight decay, and gradient-norm clipping at 10.

4 EXPERIMENTS

We conduct our experiments to show that G^{2}WAM (i) achieves state-of-the-art trajectory prediction and, with a stronger localization method, holds sufficiently high potential for even better prediction; (ii) retains performance comparable to the baselines even when transferred zero-shot to a new scene where the spatial context is lost; (iii) performs effective closed-loop navigation; (iv) benefits substantially from the proposed spatial-context mechanism; and (v) attains state-of-the-art performance together with high runtime efficiency. We first describe the setup, then address each claim in turn.

4.1 EXPERIMENTAL SETUP

Datasets. We evaluate 8-step trajectory prediction on RECON (Shah et al., 2021b), SACSoN (Hirose et al., 2024), TartanDrive (Triest et al., 2022), and SCAND (Karnan et al., 2022), using 100 fixed test queries per dataset shared across methods, following Bar et al. (2025). We further evaluate zero-shot transfer to Go Stanford (Hirose et al., 2019). For closed-loop ImageGoal navigation, we use 100 fixed Gibson (Xia et al., 2018) validation episodes from the known Mosquito scene in Habitat-Sim (Savva et al., 2019). These episodes are disjoint from the same-scene trajectories used to fine-tune the navigation model and train the localization interface. We also present a variant of our approach that uses the ground-truth pose to provide spatial context, denoted as G^{2}WAM-GT, to show how much our method can benefit from an ideal spatial context.

Baselines. The trajectory comparison includes direct navigation policies GNM (Shah et al., 2023a) and NoMaD (Sridhar et al., 2024), predictive world models NWM (Bar et al., 2025), DINO-WM (Zhou et al., 2025), DreamerV3 (Hafner et al., 2025), LeWM (Maes et al., 2026), and LS-NWM (Zhang et al., 2025a), and geometry-aware models EvoWorld (Wang et al., 2026a) and Geometry Forcing (Wu et al., 2026). Each comparison uses identical held-out queries or episodes and the same metric definitions, while methods retain their own prediction and action-selection procedures. Trainable methods use the corresponding benchmark's training trajectories for adaptation. Spatial-perceiving WAM (Chen et al., 2026), NavWAM (Azuma et al., 2026), and WAM-Nav (Yang et al., 2026) are not quantitatively compared because their official implementations were unavailable at the time of submission.

Metrics. For trajectory prediction, we report absolute trajectory error (ATE), the mean Euclidean distance between predicted and ground-truth terminal positions after eight steps, and relative pose

Table 1: 8-step trajectory prediction. Bold denotes the best and underlined the second-best value among non-GT methods.

Method	RECON	SACSoN	TartanDrive	SCAND	Average
GNM (Shah et al., 2023a)	1.87/0.73	3.71/1.00	6.65/1.62	2.12/1.61	3.59/1.24
NoMaD (Sridhar et al., 2024)	1.95/0.53	3.73/0.96	6.32/1.31	2.24/0.49	3.56/0.82
NWM (Bar et al., 2025)	**1.13**/0.35	4.12/0.96	5.63/1.18	1.28/0.33	3.04/0.71
DINO-WM (Zhou et al., 2025)	2.33/0.54	5.39/1.10	5.00/1.06	1.87/0.42	3.65/0.78
DreamerV3 (Hafner et al., 2025)	1.88/**0.34**	4.86/1.01	5.96/1.24	1.38/0.34	3.52/0.73
LeWM (Maes et al., 2026)	2.20/0.66	4.01/1.05	4.93/1.31	2.16/0.58	3.33/0.90
LS-NWM (Zhang et al., 2025a)	1.60/0.42	4.29/0.99	5.50/1.28	1.26/0.33	3.16/0.76
EvoWorld (Wang et al., 2026a)	3.10/0.72	5.86/1.22	6.55/1.35	2.95/0.63	4.61/0.98
Geometry Forcing (Wu et al., 2026)	1.64/0.48	4.47/0.94	5.23/1.11	**1.15/0.31**	3.13/0.71
G^{2}WAM	1.38/0.36	**1.18/0.40**	**2.70/0.63**	2.04/0.45	**1.82/0.46**
G^{2}WAM-GT	0.46/0.18	0.94/0.35	1.13/0.36	0.57/0.23	0.78/0.28

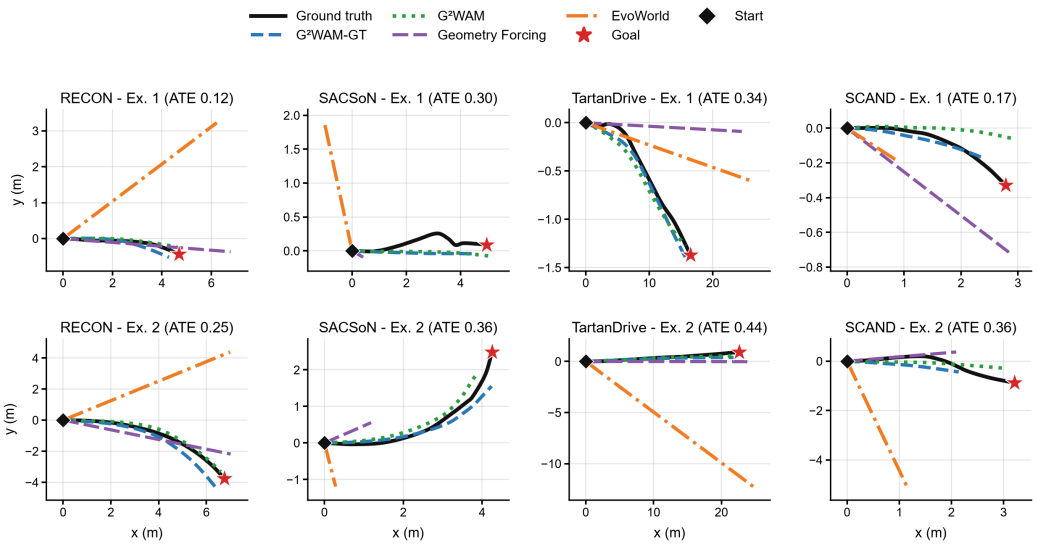


Figure 3: 8-step trajectory rollouts on RECON, SACSoN, TartanDrive, and SCAND, with all paths aligned at the rollout origin.

error (RPE), the mean per-step Euclidean error between predicted and ground-truth waypoint increments. For navigation, we report For navigation, we report success rate (SR), success weighted by path length (SPL), and final geodesic distance to goal (DTG). Each episode executes at least one action, after which the evaluator issues STOP when the geodesic distance falls below 2 m, following (Zhang et al., 2025a). For inference efficiency, we measure single-step and 16-step rollout latency on an H800 using bfloat16 and batch size 16.

4.2 Trajectory Prediction

This experiment tests claims (i) and (ii) in the prediction setting. As shown in Table 1, G^{2}WAM quantitatively attains the best average ATE/RPE of $1.82/0.46$ across the four datasets, substantially reducing ATE/RPE by $40.1\%/35.2\%$ relative to the second-best baseline. The improvement is most pronounced on SACSoN and TartanDrive, which we attribute to the reliable spatial and motion conditions in SACSoN and the diverse motion scales in TartanDrive. In addition, our method performs comparably to the baselines on RECON but degrades on SCAND, mainly because cross-platform motion, operator variation, and dynamic pedestrians severely limit localization performance, thereby undermining the gains it provides to the world model. Note that, when using ground-truth pose, G^{2}WAM-GT achieves a substantial performance gain. This gain indicates that our method holds high potential when provided with a better spatial context.

Table 2: Zero-shot trajectory prediction on Go Stanford. Bold denotes the best result and underlining the second-best result.

Method	ATE ↓	RPE ↓
NWM (Bar et al., 2025)	2.3535	<u>0.6447</u>
DINO-WM (Zhou et al., 2025)	2.8993	0.7275
LeWM (Maes et al., 2026)	3.0258	0.7602
Geometry Forcing (Wu et al., 2026)	**2.0981**	**0.6063**
G^{2}WAM	<u>2.3096</u>	0.6510
G^{2}WAM-PoseFT	1.9866	0.5434
G^{2}WAM-GT	1.5424	0.4296

Table 3: Gibson image goal navigation. Bold denotes the best result and underlining the second-best result.

Method	SR ↑	SPL ↑	DTG ↓
GNM (Shah et al., 2023a)	24.0%	22.8%	5.37
NoMaD (Sridhar et al., 2024)	28.0%	27.4%	4.78
NWM (Bar et al., 2025)	32.0%	26.35%	5.39
DINO-WM (Zhou et al., 2025)	<u>54.0%</u>	<u>36.4%</u>	4.56
DreamerV3 (Hafner et al., 2025)	44.0%	33.7%	5.04
LeWM (Maes et al., 2026)	42.0%	33.01%	<u>4.51</u>
G^{2}WAM	**63.0%**	**59.2%**	**3.39**
G^{2}WAM-GT	71.0%	70.0%	3.04

The qualitative rollouts in Fig. 3 illustrate the same trend. Baseline trajectories often drift away from the target direction or accumulate scale error over the 8-step horizon, whereas G^{2}WAM rollouts remain better aligned with the ground-truth path. G^{2}WAM-GT yields the cleanest rollouts, and G^{2}WAM preserves much of this structure under an image-based pose interface. These results support the main design choice of predicting canonical waypoint increments directly from a goal-conditioned latent transition, rather than relying on a separate sampling or trajectory-selection stage.

Table 2 evaluates our method when zero-shot transferred to a new scene without spatial context. Though not the primary scenario of this paper, this setting tests whether our world action model backbone has learned generalizable navigation priors without spatial context. We also report results with a localization module trained on a subset of the dataset, referred to as G^{2}WAM-PoseFT, and with ground-truth pose simulating external localization devices such as GPS or UWM. Without spatial context, our method performs comparably to the baselines, indicating that our WAM does not rely solely on spatial context but possesses certain navigation priors. With pose assistance, it improves ATE/RPE by 14.0%/16.5%.

4.3 Closed-Loop Navigation

This experiment tests claim (iii) in the closed-loop navigation. Concretely, it tests whether the predicted waypoints remain useful after their execution changes the observation and the estimated spatial context, rather than only achieving low open-loop trajectory error. Table 3 shows that G^{2}WAM achieves 63.0% SR, 59.2% SPL, and 3.39 m final DTG. Compared with DINO-WM, the strongest baseline, it improves the two metrics by 9.0 and 22.8 percentage points, respectively. Its final DTG is also 25% lower than that of LeWM, the strongest baseline on this metric. The improvements across all three measures show that the waypoint predictions remain effective when executed in a repeated observe-and-replan loop, not only in open-loop trajectory prediction. The larger SPL gain over the baselines indicates that, aided by spatial context, our method navigates more purposefully and thus more efficiently. G^{2}WAM-GT reaches 71.0% SR, 70.0% SPL, and 3.04 m final DTG. The gaps to G^{2}WAM indicate that our method still leaves considerable room for improvement. Figure 4 shows the navigation success rate under different initial start-to-goal distances. The results indicate that spatial context plays a substantial role in both short- and long-range navigation. For long-range navigation, our spatial context does not yield a larger gap over the baselines, possibly because longer distances offer more opportunities for complex structured obstacles to appear between the current position and the goal, which makes direct goal-directed guidance insufficient.

4.4 Ablations

This section tests claim (iv): the proposed spatial-context mechanism markedly improves navigation. We ablate the spatial conditions to assess whether goal-relative pose and causal motion provide complementary information for trajectory prediction.

We evaluate the method under different conditioning settings, as in Table 4. The variant without any spatial context is denoted as *None*; the variant that provides only the goal and current-image poses is denoted as *Pose only*; and the variant that additionally provides the goal, current, and history trajectories is denoted as *Pose+motion*. For *Pose+motion*, we test three combinations using either ground-truth or R-SCoRe-estimated poses, where, e.g., *GT+GT* indicates that both pose and motion

Table 4: Conditioning ablation under the 8-step trajectory-prediction protocol.

Dataset	Pose + motion			Pose only		None
	GT+GT	Pred.+GT	Pred.+Pred.	GT	Pred.	
RECON	0.46/0.18	1.17/0.33	1.38/0.36	0.77/0.33	1.38/0.37	2.02/0.58
SACSoN	0.94/0.35	1.15/0.39	1.18/0.40	1.42/0.48	1.42/0.46	2.32/0.62
TartanDrive	1.13/0.36	1.70/0.48	2.70/0.63	3.56/0.96	4.53/0.93	5.64/1.19
SCAND	0.57/0.23	1.55/0.38	2.04/0.45	0.78/0.29	1.96/0.44	1.96/0.47
Average	0.78/0.28	1.39/0.39	1.82/0.46	1.63/0.52	2.32/0.55	2.99/0.71

Table 5: Runtime analysis on an NVIDIA H800 PCIe GPU. Step and 16-step are model-side timings.

Type	Method	1-step (ms) ↓	16-step (ms) ↓	Action selection (s) ↓
L-WM	DreamerV3	9	**33**	0.65
	LeWM	**2**	<u>34</u>	**0.36**
	DINO-WM	26	423	14.45
I-WM	NWM	705	11430	10.98
	Geometry Forcing	23486	47227	541.23
G-WM	EvoWorld	7048	91175	150.26
G-WAM	G^{2}WAM	<u>4</u>	47	<u>0.51</u>

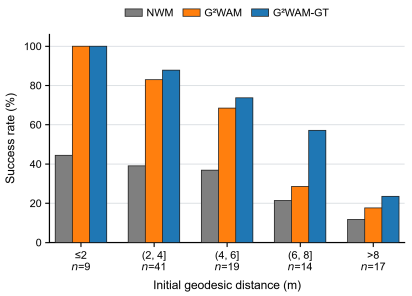


Figure 4: Closed-loop success rate grouped by initial geodesic distance. n denotes the number of episodes in each distance range.

use ground truth. The results show that both inputs are necessary. With the current state-of-the-art R-SCoRe pipeline, pose and motion yield average gains of 39.1%/35.2% on ATE/RPE, respectively, and with ideal pose estimation or external localization these gains can expand to 73.9%/60.6%.

4.5 Runtime Analysis

To assess the high computational efficiency of our approach as claimed in (v), we compare inference latency under a 16-step rollout and action-selection protocol, as shown in Table 5. We annotate each method with its type: V-WM denotes a video world model, L-WM a latent world model that rolls out in latent space, G-WM a geometry-aware world model, and G-WAM a geometry-aware world-action model. No world action model methods are included, since the relevant works Chen et al. (2026); Azuma et al. (2026); Yang et al. (2026) are not open-sourced. World-model-based methods generally take longer because they must perform CEM planning. Our method is comparable in efficiency to state-of-the-art L-WM methods and has a clear advantage in action selection. We further analyze the runtime breakdown of action selection. Our method's runtime mainly consists of localization time and world action model inference time, where localization takes 0.42 s, which also offers a clear advantage over EvoWorld's geometric-context encoding time of 0.57 s. EvoWorld's overall action-selection cost reaches 8.7 s, since it re-encodes the geometric context at every CEM rollout.

5 Conclusion

We presented G^{2}WAM, a geometry-grounded world action model for image-goal navigation. G^{2}WAM encodes spatial cues through a learned end-to-end localization module that anchors the goal and observed frames in a common reference frame, yielding a stable and consistent spatial context that enables efficient navigation without explicit 3D reconstruction. We adopt the world-action-model paradigm and move the rollout into latent space, jointly predicting future actions and latent future states while avoiding costly CEM-style planning and pixel-level reconstruction. Extensive experiments show that our method achieves state-of-the-art trajectory prediction and navigation performance, all with high inference efficiency. Future work will focus on extending G^{2}WAM to longer-horizon navigation and broader embodied tasks.

REFERENCES

Daichi Azuma, Taiki Miyanishi, Koya Sakamoto, Shuhei Kurita, Yaonan Zhu, Petr Khrapchenkov, Motoaki Kawanabe, Yusuke Iwasawa, and Yutaka Matsuo. NavWAM: A navigation world action model for goal-conditioned visual navigation. *arXiv preprint arXiv:2606.13494*, 2026.

Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 15791–15801, 2025 .

Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. DSAC—differentiable RANSAC for camera localization. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017 .

Hong Chen, Daqi Liu, Zehan Zhang, Haiguang Wang, Tianhao Lu, Longfei Yan, Haiyang Sun, Fangzhen Li, Hongwei Xie, Bing Wang, Guang Chen, Hangjun Ye, and Yihua Tan. Pondering the way: Spatial-perceiving world action model for embodied navigation. In *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2026 .

Yijie Deng, Shuaihang Yuan, and Yi Fang. AnyImageNav: Any-view geometry for precise last-meter image-goal navigation. *arXiv preprint arXiv:2604.05351*, 2026.

Paul Furgale and Timothy D. Barfoot. Visual teach and repeat for long-range rover autonomy. *Journal of Field Robotics*, 27(5):534–560, 2010 .

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025 .

Noriaki Hirose, Fei Xia, Roberto Martin-Martin, Amir Sadeghian, and Silvio Savarese. Deep visual MPC-policy learning for navigation. *IEEE Robotics and Automation Letters*, 4(4):3184–3191, 2019 .

Noriaki Hirose, Dhruv Shah, Ajay Sridhar, and Sergey Levine. SACSoN: Scalable autonomous control for social navigation. *IEEE Robot. Autom. Lett.*, 9(1):49–56, 2024 .

Toru Ishikawa and Daniel R. Montello. Spatial knowledge acquisition from direct experience in the environment: Individual differences in the development of metric knowledge and the integration of separately learned places. *Cognitive Psychology*, 52(2):93–129, 2006 .

Xudong Jiang, Fangjinhua Wang, Silvano Galliani, Christoph Vogel, and Marc Pollefeys. R-SCoRe: Revisiting scene coordinate regression for robust large-scale visual localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11536–11546, 2025.

Haresh Karnan, Anirudh Nair, Xuesu Xiao, Garrett Warnell, Sören Pirk, Alexander Toshev, Justin Hart, Joydeep Biswas, and Peter Stone. Socially compliant navigation dataset (SCAND): A large-scale dataset of demonstrations for social navigation. *IEEE Robot. Autom. Lett.*, 7(4):11807–11814, 2022.

Po-Chien Luan, Zimin Xia, Wuyang Li, Yang Gao, and Alexandre Alahi. Drift-resistant navigation world model with anchored epipolar guidance. *arXiv preprint arXiv:2605.24761*, 2026.

Yueen Ma, Zenglin Xu, and Irwin King. 4DGS-WAM: Bridging past and future with an object-centric world action model based on 4D gaussian splatting. *arXiv preprint arXiv:2608.25956*, 2026.

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

Yuta Oshima, Yusuke Iwasawa, Masahiro Suzuki, Yutaka Matsuo, and Hiroki Furuta. WorldPack: Dynamic frame compression for long-context video world modeling. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856 .

Ruslan Partsey, Erik Wijmans, Naoki Yokoyama, Oles Dobosevych, Dhruv Batra, and Oleksandr Maksymets. Is mapping necessary for realistic PointGoal navigation? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17232–17241, 2022 .

Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A. Efros, and Trevor Darrell. Zero-shot visual imitation. In *International Conference on Learning Representations*, 2018 .

Xuanchi Ren, Tianchang Shen, Jiahui Huang, Huan Ling, Yifan Lu, Merlin Nimier-David, Thomas Müller, Alexander Keller, Sanja Fidler, and Jun Gao. GEN3C: 3D-informed world-consistent video generation with precise camera control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6121–6132, 2025 .

Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 4938–4947, 2020 .

Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. In *International Conference on Learning Representations*, 2018 .

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 *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pp. 9339–9347, 2019 .

Dhruv Shah, Benjamin Eysenbach, Gregory Kahn, Nicholas Rhinehart, and Sergey Levine. ViNG: Learning open-world navigation with visual goals. In *IEEE International Conference on Robotics and Automation*, pp. 13215–13222, 2021a .

Dhruv Shah, Benjamin Eysenbach, Nicholas Rhinehart, and Sergey Levine. Rapid exploration for open-world navigation with latent goal models. In *Proc. Conf. Robot Learn. (CoRL)*, volume 164 of *Proceedings of Machine Learning Research*, pp. 674–684. PMLR, 2021b.

Dhruv Shah, Ajay Sridhar, Arjun Bhorkar, Noriaki Hirose, and Sergey Levine. GNM: A general navigation model to drive any robot. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 7226–7233, 2023a .

Dhruv Shah, Ajay Sridhar, Nitish Dashora, Kyle Stachowicz, Kevin Black, Noriaki Hirose, and Sergey Levine. ViNT: A foundation model for visual navigation. In *Proc. Conf. Robot Learn. (CoRL)*, volume 229 of *Proceedings of Machine Learning Research*, pp. 711–733. PMLR, 2023b.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël 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, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3. *arXiv preprint arXiv:2508.10104*, 2025.

Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. NoMaD: Goal masked diffusion policies for navigation and exploration. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 63–70, 2024 .

Samuel Triest, Matthew Sivaprakasam, Sean J. Wang, Wenshan Wang, Aaron M. Johnson, and Sebastian Scherer. TartanDrive: A large-scale dataset for learning off-road dynamics models. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 2546–2552, 2022 .

Jiahao Wang, Luoxin Ye, Taiming Lu, Junfei Xiao, Jiahan Zhang, Yuxiang Guo, Xijun Liu, Rama Chellappa, Cheng Peng, Alan Yuille, and Jieneng Chen. EvoWorld: Evolving panoramic world generation with explicit 3D memory. In *ICLR 2026 Workshop on World Models*, 2026a . Workshop paper.

Ranxiao Frances Wang and Elizabeth S. Spelke. Human spatial representation: Insights from animals. *Trends in Cognitive Sciences*, 6(9):376–382, 2002 .

Weijie Wang, Haoyu Zhao, Yifan Yang, Feng Chen, Zeyu Zhang, Yefei He, Zicheng Duan, Donny Y. Chen, Yuqing Yang, and Bohan Zhuang. Latent spatial memory for video world models. *arXiv preprint arXiv:2606.09828*, 2026b.

William H. Warren. Non-euclidean navigation. *Journal of Experimental Biology*, 222(Suppl$_1$) : $jeb187971$, 2019.

Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. DD-PPO: Learning near-perfect PointGoal navigators from 2.5 billion frames. In *International Conference on Learning Representations*, 2020 .

Haoyu Wu, Diankun Wu, Tianyu He, Junliang Guo, Yang Ye, Yueqi Duan, and Jiang Bian. Geometry forcing: Marrying video diffusion and 3D representation for consistent world modeling. In *The Fourteenth International Conference on Learning Representations*, 2026 .

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

Fei Xia, Amir R. Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 9068–9079, 2018 .

Ning Yang, Yan Huang, Kaiwen Peng, Ziheng He, Kai Wang, Cui Miao, Kailin Lyu, Guo Li, Xiaofeng Wang, Zheng Zhu, Jing Liu, and Nianfeng Liu. WAM-Nav: Asymmetric latent world-action modeling for unified visual navigation. *arXiv preprint arXiv:2606.04907*, 2026.

Zhiwei Zhang, Hui Zhang, Xieyuanli Chen, Kaihong Huang, Chenghao Shi, and Huimin Lu. Latent-space autoregressive world model for efficient and robust image-goal navigation. *arXiv preprint arXiv:2511.11011*, 2025a.

Zhiwei Zhang, Hui Zhang, Kaihong Huang, Chenghao Shi, and Huimin Lu. Efficient image-goal navigation with representative latent world model. *arXiv preprint arXiv:2511.11011*, 2025b.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Proc. Int. Conf. Mach. Learn. (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 79115–79135. PMLR, 2025.

A Implementation Details

A.1 World-Action Model Architecture

Visual encoding. All current, historical, and goal images are resized to 224×224 before being passed through a frozen DINOv3 encoder (Siméoni et al., 2025). The encoder produces patch-level features with latent dimension $D = 384$. We retain the patch structure rather than spatially pooling the observation features, because the transition model uses patch-level information to represent spatial changes across observations.

At planning step t, the model receives the four most recent observation latents, $\boldsymbol{z}_{t-3:t} \in \mathbb{R}^{4\times P\times D}$, and the goal-image latent $\boldsymbol{z}_g \in \mathbb{R}^{P\times D}$. The goal latent is first layer-normalized and then mean-pooled over its P patches. A two-layer MLP with a GELU activation maps the pooled feature into the goal embedding:

$$\boldsymbol{q}_g = \phi_g\left(\mathrm{pool}\left(\mathrm{LN}(\boldsymbol{z}_g)\right)\right). \tag{8}$$

The DINOv3 encoder remains frozen during world-action-model training, while the goal projection, transition module, conditioners, joint decoder, and output heads are optimized.

Relative-pose embedding. The relative-pose conditioner receives

$$\boldsymbol{p}_t = (\Delta x_t^g, \Delta y_t^g, \sin \Delta\theta_t^g, \cos \Delta\theta_t^g)\,, \tag{9}$$

where the planar translation is expressed in the current robot frame and divided by the dataset waypoint spacing. The implementation separates the translation and yaw components before embedding them.

Let

$$d_t = \sqrt{(\Delta x_t^g)^2 + (\Delta y_t^g)^2}, \qquad \boldsymbol{u}_t = \frac{(\Delta x_t^g, \Delta y_t^g)}{\max(d_t, \epsilon)}, \tag{10}$$

where $\boldsymbol{u}_t$ is the normalized goal direction and ϵ prevents division by zero. The translation branch receives $[\boldsymbol{u}_t, \log(1 + d_t)]$, while the yaw branch receives $(\sin \Delta\theta_t^g, \cos \Delta\theta_t^g)$. Each branch is a two-layer MLP with a GELU activation. Their outputs are concatenated and passed through another two-layer MLP followed by layer normalization:

$$\boldsymbol{e}_t^{\text{pose}} = \phi_{\text{fuse}}\left([\phi_{\text{trans}}\left(\boldsymbol{u}_t, \log(1 + d_t)\right); \phi_{\text{yaw}}\left(\sin \Delta\theta_t^g, \cos \Delta\theta_t^g\right)]\right). \tag{11}$$

This decomposition preserves goal direction, logarithmic distance, and relative yaw rather than treating the four-dimensional pose condition as an unstructured vector.

Causal-motion embedding. The motion conditioner receives the five-dimensional profile

$$\boldsymbol{m}_t = \left[\bar{d}_t, \bar{\theta}_t, d_s, \theta_s, \ell\right]^{\mathsf{T}}, \tag{12}$$

where $\bar{d}_t$ and $\bar{\theta}_t$ summarize recent estimated translation and rotation, d_s and θ_s are dataset-level translation and yaw scales, and ℓ is the waypoint spacing in metres. All entries are nonnegative. The implementation applies a componentwise $\log(1 + x)$ transform before passing the profile through a two-layer MLP with a GELU activation and layer normalization:

$$\boldsymbol{e}_t^{\text{motion}} = \phi_{\text{motion}}\left(\log(\mathbf{1} + \boldsymbol{m}_t)\right). \tag{13}$$

The logarithmic transform reduces the effect of scale differences between datasets and platforms.

Spatiotemporal latent transition. The four-frame latent history is first layer-normalized and reshaped from $\mathbb{R}^{H \times P \times D}$ into a sequence containing HP patch tokens. Learned temporal and patch positional embeddings are added to each token:

$$\boldsymbol{x}_{i,p} = \text{LN}(\boldsymbol{z}_{i,p}) + \boldsymbol{e}_i^{\text{time}} + \boldsymbol{e}_p^{\text{patch}}, \tag{14}$$

where i indexes the history frame and p indexes the patch. Projected goal and relative-pose embeddings are then added to every spatiotemporal token:

$$\tilde{\boldsymbol{x}}_{i,p} = \boldsymbol{x}_{i,p} + \phi_{g \to \text{tok}}(\boldsymbol{q}_g) + \phi_{p \to \text{tok}}(\boldsymbol{e}_t^{\text{pose}}). \tag{15}$$

A token-selection module compresses the full spatiotemporal sequence into eight learned tokens. Concretely, it predicts a normalized attention map over all HP input patches for each learned token and uses the resulting attention weights to compute weighted sums of the input features. The eight selected tokens therefore summarize the four-frame latent history while retaining information from all spatial locations.

The goal and relative-pose projections are added again to the learned tokens before they enter a four-layer Transformer encoder. Each Transformer layer uses eight attention heads and a feed-forward dimension of $4D$. The encoded tokens are subsequently fused back into the original spatiotemporal patch sequence through a learned token-to-patch mixing module. This operation produces a fused feature for every patch in every history frame rather than returning only a small set of global tokens.

The fused patches corresponding to the latest observation are used as queries in an eight-head cross-attention operation. The complete fused spatiotemporal sequence serves as keys and values:

$$\boldsymbol{h}_{t+k} = \text{MHA}\left(\boldsymbol{x}^{\text{last}}, \boldsymbol{x}^{\text{fused}}, \boldsymbol{x}^{\text{fused}}\right). \tag{16}$$

The attended features are combined with the latest-frame patches, a projected goal bias, and a projected relative-pose bias:

$$\boldsymbol{c}_{t+k} = \text{LN}\left(\boldsymbol{h}_{t+k} + \boldsymbol{x}^{\text{last}} + \phi_{g \to \text{next}}(\boldsymbol{q}_g) + \phi_{p \to \text{next}}(\boldsymbol{e}_t^{\text{pose}})\right). \tag{17}$$

The resulting $\boldsymbol{c}_{t+k} \in \mathbb{R}^{P\times D}$ is the patch-level transition context used by both latent-state and action prediction.

The relative-pose embedding is therefore injected at four locations: the full patch sequence, the learned-token sequence, the cross-attention residual, and the pooled step state. The causal-motion embedding is injected only into the pooled step state. This asymmetry lets the relative pose constrain the spatial evolution of the latent representation, while recent motion and platform scale primarily affect action generation.

Joint latent–action decoder. The patch-level context is mean-pooled and combined with the pose and motion embeddings:

$$\boldsymbol{s}_{t+k} = \mathrm{pool}(\boldsymbol{c}_{t+k}) + \boldsymbol{e}_t^{\mathrm{pose}} + \boldsymbol{e}_t^{\mathrm{motion}}. \tag{18}$$

This produces a compact $\boldsymbol{s}_{t+k} \in \mathbb{R}^D$ step representation for control.

The joint decoder contains learned action and state-summary query tokens. The shared step representation is added to both queries. Patch-level latent tokens are formed by adding a learned latent-type embedding to $\boldsymbol{c}_{t+k}$. Action, state-summary, and latent-type embeddings distinguish the three token streams. The resulting token sequence is processed by a two-layer Transformer encoder with eight attention heads, pre-normalization, GELU activations, and a feed-forward dimension of $4D$.

Separate linear heads read the transformed action query and latent patch tokens. The action head produces the canonical waypoint increment

$$\hat{\boldsymbol{a}}_{t+k}^{\mathrm{can}} = (\Delta x_{t+k}, \Delta y_{t+k}, \Delta\theta_{t+k}), \tag{19}$$

while the latent head produces the patchwise candidate $\boldsymbol{z}_{t+k+1}^{\mathcal{J}}$. The implementation also retains an internal state-summary output from the state query, while the recurrent latent update and executable action are determined by the latent and action outputs.

Residual latent update. In parallel with the joint decoder, a linear base projection produces

$$\boldsymbol{z}_{t+k+1}^{\mathrm{B}} = \phi_{\mathrm{B}}(\boldsymbol{c}_{t+k}). \tag{20}$$

The final next-step latent combines the joint-decoder prediction and the base projection:

$$\hat{\boldsymbol{z}}_{t+k+1} = \mathrm{LN}\left(\alpha_{\mathcal{J}}\boldsymbol{z}_{t+k+1}^{\mathcal{J}} + \alpha_{\mathrm{B}}\boldsymbol{z}_{t+k+1}^{\mathrm{B}}\right), \tag{21}$$

where $\alpha_{\mathcal{J}} = 1.0$ and $\alpha_{\mathrm{B}} = 0.1$. The base branch provides a direct patch-level transition, while the joint branch supplies the dominant state-action prediction.

The predicted latent is appended to the fixed-length history and the oldest latent is removed:

$$\boldsymbol{z}_{t+k-H+2:t+k+1} = [\boldsymbol{z}_{t+k-H+2:t+k}; \hat{\boldsymbol{z}}_{t+k+1}]. \tag{22}$$

The model repeats this procedure autoregressively for eight rollout steps. Goal, pose, and motion conditions remain fixed within one rollout and are recomputed after the next observation in the closed-loop navigation cycle. No future RGB image is reconstructed at any rollout step.

Canonical-motion adaptation. The joint decoder predicts waypoint increments in a canonical motion space. The canonical-motion adapter converts these predictions into the normalized action interface used by each dataset or platform. The planar components are multiplied by the recent translation scale, while the yaw component is multiplied by the dataset-level yaw scale:

$$\boldsymbol{a}_{t+k,xy}^{\mathrm{raw}} = \tilde{d}_t\hat{\boldsymbol{a}}_{t+k,xy}^{\mathrm{can}}, \qquad a_{\theta,t+k}^{\mathrm{raw}} = \tilde{\theta}_s\hat{a}_{\theta,t+k}^{\mathrm{can}}, \tag{23}$$

where

$$\tilde{d}_t = \mathrm{clip}(\bar{d}_t, 0.1, 4.0), \qquad \tilde{\theta}_s = \mathrm{clip}(\theta_s, 0.05, 1.0). \tag{24}$$

The scaled planar action is normalized using fixed bounds $\boldsymbol{b}_{\min} = (-2.5, -4.0)$ and $\boldsymbol{b}_{\max} = (5.0, 4.0)$:

$$\mathcal{N}_{xy}(\boldsymbol{\xi}) = 2\frac{\boldsymbol{\xi} - \boldsymbol{b}_{\min}}{\boldsymbol{b}_{\max} - \boldsymbol{b}_{\min}} - \mathbf{1}. \tag{25}$$

The final action output is

$$\hat{\boldsymbol{a}}_{t+k} = \left[\mathcal{N}_{xy}\left(\tilde{d}_t\hat{\boldsymbol{a}}_{t+k,xy}^{\mathrm{can}}\right); \tilde{\theta}_s\hat{a}_{\theta,t+k}^{\mathrm{can}}\right]. \tag{26}$$

The relative pose thus supplies target geometry, while the motion profile determines how a canonical waypoint is converted into dataset- or platform-specific executable motion.

A.2 Training Details

Training objective. The localization interface is trained separately from the world-action model. The world-action model is optimized using the multi-term objective in Eq. 6. The latent loss combines patchwise cosine distance with a smaller mean-squared-error term:

$$\mathcal{L}_z = 1 - \mathrm{Cos}\left(\hat{\boldsymbol{Z}}, \boldsymbol{Z}^*\right) + 0.3\,\mathrm{MSE}\left(\hat{\boldsymbol{Z}}, \boldsymbol{Z}^*\right). \tag{27}$$

The action loss applies a smooth-ℓ_1 loss to the canonical translation and yaw increments. Turning steps receive larger weights to discourage collapse to an always-forward prediction:

$$\mathcal{L}_a = \frac{\sum_{b,k} w_{b,k} \ell_{\mathrm{SL1}}\left(\hat{\boldsymbol{a}}^{\mathrm{can}}_{b,t+k}, \boldsymbol{a}^{\mathrm{can},*}_{b,t+k}\right)}{\sum_{b,k} w_{b,k}}. \tag{28}$$

The trajectory and yaw losses compare cumulative executable translations and rotations over the complete rollout. Planar action increments are converted back from their normalized representation before accumulation.

The loss coefficients are

$$\lambda_z = 0.3, \qquad \lambda_a = 3.0, \qquad \lambda_{\mathrm{traj}} = 2.0, \qquad \lambda_{\mathrm{yaw}} = 0.5. \tag{29}$$

Optimization settings. We use AdamW with batch size 32, zero weight decay, and gradient-norm clipping at 10. The four-dataset checkpoint is trained for 25,800 parameter updates. It is subsequently adapted to Gibson using a mixture of expert, long-goal imitation, and recovery trajectories with a learning rate of 5×10^{-6}.

For the reported Mosquito closed-loop experiment, this Gibson-adapted checkpoint is further fine-tuned for 500 optimization steps on 200 Mosquito trajectories using a learning rate of 2×10^{-6}. These 200 trajectories are disjoint from the fixed 100 evaluation episodes. They are also used as the reference observations for constructing the scene-specific localization database; consequently, the experiment follows a known-scene but held-out-trajectory protocol. No evaluation episode is used for world-action-model fine-tuning or localization training.

The DINOv3 encoder remains frozen throughout training and fine-tuning. The localization interface is trained independently and is not updated by the world-action-model losses. At inference time, it supplies only the relative-pose and causal-motion conditions.

A.3 Localization Interface

R-SCoRe localization. The main text defines f_{loc} as an interface that maps an image to a pose in the shared scene frame (Eq. 1). We instantiate this interface with R-SCoRe (Jiang et al., 2025), a learned scene-coordinate localization method. R-SCoRe does not directly regress the final camera pose. Instead, it combines local image descriptors with retrieved global scene encodings to predict three-dimensional scene coordinates for query keypoints. A geometric PnP–RANSAC solver then recovers the camera pose from the resulting 2D–3D correspondences.

The original R-SCoRe evaluation reports a 64.3% localization success rate within $(0.25\,\mathrm{m}, 2^\circ)$ and an inference time of approximately 140–270 ms per query on an RTX 2080. These values describe the localization benchmarks and hardware used in the original R-SCoRe evaluation, rather than the runtime of our Habitat implementation. The model-side latency reported in the main paper excludes localization, simulation, and action execution.

Reference-view retrieval. For a query image $\boldsymbol{I}_j$, NetVLAD produces a global descriptor and retrieves a set of candidate reference views from the scene database $\mathcal{D}_{\mathrm{S}}$:

$$\mathcal{H}_j = \mathrm{Retrieve}_K\left(\mathrm{NetVLAD}(\boldsymbol{I}_j), \mathcal{D}_{\mathrm{S}}\right). \tag{30}$$

The Habitat implementation uses product-quantized nearest-neighbor search over the stored NetVLAD descriptors. The retrieved view index $r \in \mathcal{H}_j$ selects a candidate-specific global scene encoding $\boldsymbol{g}_r$ from the R-SCoRe reference database.

The NetVLAD descriptor is used only to retrieve candidate views. It is distinct from the global scene encoding $\boldsymbol{g}_r$ used by the scene-coordinate prediction network.

Scene-coordinate prediction. A local feature encoder extracts query keypoints $\boldsymbol{u}$ and their descriptors $\boldsymbol{d}_{j,\boldsymbol{u}}$. For each candidate reference view r, its global scene encoding is concatenated with every local query descriptor. The scene-coordinate head then predicts a three-dimensional scene coordinate:

$$\hat{\boldsymbol{X}}_{j,\boldsymbol{u}}^{(r)} = f_{\mathrm{SCR}}\left([\boldsymbol{d}_{j,\boldsymbol{u}}; \boldsymbol{g}_r]\right). \tag{31}$$

Because the global encoding is candidate-dependent, the same query keypoint produces one scene-coordinate hypothesis for every retrieved candidate. This allows the localization system to evaluate multiple possible regions of the known scene.

Pose estimation and candidate selection. A separate PnP–RANSAC pose hypothesis is estimated for every retrieved candidate:

$$\left(\hat{\boldsymbol{T}}_{C_j\leftarrow \mathrm{S}}^{(r)}, \mathcal{I}_j^{(r)}\right) = \mathrm{PnPRANSAC}\left(\{(\boldsymbol{u}, \hat{\boldsymbol{X}}_{j,\boldsymbol{u}}^{(r)})\}_{\boldsymbol{u}}, \boldsymbol{K}_j\right), \tag{32}$$

where $\boldsymbol{K}_j$ is the query-camera intrinsic matrix and $\mathcal{I}_j^{(r)}$ is the corresponding RANSAC inlier set. The selected candidate is

$$r^* = \arg\max_{r\in\mathcal{H}_j} \left|\mathcal{I}_j^{(r)}\right|. \tag{33}$$

Thus, candidate selection is based on geometric consistency rather than retrieval similarity alone.

The PnP solver returns a scene-to-camera pose. We invert it to obtain the camera-to-scene pose and then apply the known camera-to-robot extrinsic calibration:

$$\hat{\boldsymbol{T}}_{B_j}^{\mathrm{S}} = \hat{\boldsymbol{T}}_{C_j}^{\mathrm{S}} \left(\boldsymbol{T}_{B\leftarrow C}\right)^{-1}. \tag{34}$$

The resulting calibrated robot pose is the $\hat{\boldsymbol{T}}_j^{\mathrm{S}}$ used in Eq. 1.

Shared-frame spatial context. The current, goal, and history images are independently localized in the same scene frame. Their relative transformations therefore provide the goal-relative pose and consecutive history motions:

$$\hat{\boldsymbol{T}}_{t\to g} = \left(\hat{\boldsymbol{T}}_t^{\mathrm{S}}\right)^{-1} \hat{\boldsymbol{T}}_g^{\mathrm{S}}, \qquad \hat{\boldsymbol{T}}_{i\to i+1} = \left(\hat{\boldsymbol{T}}_i^{\mathrm{S}}\right)^{-1} \hat{\boldsymbol{T}}_{i+1}^{\mathrm{S}}. \tag{35}$$

These relative transformations are invariant to a common rigid transformation of the shared scene frame:

$$\left(\boldsymbol{G}\hat{\boldsymbol{T}}_t^{\mathrm{S}}\right)^{-1} \left(\boldsymbol{G}\hat{\boldsymbol{T}}_g^{\mathrm{S}}\right) = \left(\hat{\boldsymbol{T}}_t^{\mathrm{S}}\right)^{-1} \hat{\boldsymbol{T}}_g^{\mathrm{S}}, \tag{36}$$

for any $\boldsymbol{G} \in \mathrm{SE}(3)$. The world-action model is therefore independent of the arbitrary global origin and orientation chosen for the scene coordinate system. This invariance does not eliminate localization error: the current, goal, and history images must still be localized consistently in the shared frame.

The planar translation of $\hat{\boldsymbol{T}}_{t\to g}$ is converted to the model coordinate convention and divided by the waypoint spacing ℓ before forming $\boldsymbol{p}_t$. The consecutive history transformations provide the history-dependent entries of $\boldsymbol{m}_t$.

Localization modularity. The retrieved reference images, NetVLAD descriptors, local keypoint descriptors, candidate global encodings, predicted scene coordinates, PnP hypotheses, and RANSAC inlier sets are internal quantities of the localization interface. They are not provided directly to the world-action model.

The world-action model consumes only the current-to-goal relative transformation and the recent causal-motion profile. Consequently, the localization system can be replaced without changing the world-action-model architecture, provided that the replacement returns poses in a consistent metric frame. Possible alternatives include a traditional place-recognition and pose-estimation pipeline, visual SLAM, an external tracking system, or another learned relative-pose estimator.

Known-scene scope. R-SCoRe requires a scene-specific reference database and a scene-coordinate prediction model trained from reference observations. Our Habitat experiment therefore follows a known-scene protocol. The fixed evaluation episodes are disjoint from the trajectories used to construct the localization database and fine-tune the navigation model, although the reference and evaluation trajectories belong to the same scene.

This setup evaluates navigation to previously observed places from held-out trajectories rather than navigation in a completely unseen environment. Applying the same world-action model in an unseen scene would require an alternative shared-frame localization source, such as online mapping, visual SLAM, or a map-free relative-pose estimator. Once the relative-pose and motion conditions are available, the world-action-model architecture itself does not need to be changed.

Online localization sequence. The goal image is localized once at the beginning of an episode and its estimated pose is cached. Every newly observed RGB frame is localized online. Only the current and previously observed images are used; neither future images nor future actions are available to the localization interface.

For causal-motion estimation, at most four available history poses are retained. The pose history is updated only after the corresponding frame has been observed. The estimated-pose evaluation does not use ground-truth egomotion, and invalid localization estimates do not silently fall back to ground-truth poses.

Causal estimated motion. Let $\boldsymbol{r}_i$ and ψ_i denote the calibrated robot-frame position and yaw of history frame i. The recent translation scale is computed as the median nonzero displacement in waypoint units:

$$\bar{d}_t = \mathrm{median}_{i \in \mathcal{V}_t} \frac{\|\boldsymbol{r}_{i+1} - \boldsymbol{r}_i\|_2}{\ell}, \tag{37}$$

where $\mathcal{V}_t$ contains the consecutive history pairs with nonzero translation. The recent rotation scale is the mean absolute wrapped yaw change:

$$\bar{\theta}_t = \mathrm{mean}_i \left|\mathrm{atan2}\left(\sin(\psi_{i+1} - \psi_i), \cos(\psi_{i+1} - \psi_i)\right)\right|. \tag{38}$$

Before two valid history frames are available, $\bar{d}_t$ and $\bar{\theta}_t$ fall back to their dataset-level translation and yaw scales. If all available history translations are zero, $\bar{d}_t$ also falls back to the dataset-level translation scale. Its returned value is lower-bounded by 0.1. The remaining motion-profile entries are the dataset-level translation scale d_s, dataset-level yaw scale θ_s, and waypoint spacing ℓ.

Habitat coordinate conversion. For Habitat, the R-SCoRe pose engine returns a camera-to-world transformation $\boldsymbol{T}_{WC}$. The current-to-goal camera transformation is

$$\boldsymbol{T}_{C_tC_g} = \boldsymbol{T}_{WC_t}^{-1} \boldsymbol{T}_{WC_g}. \tag{39}$$

Let $\boldsymbol{t} = (t_x, t_y, t_z)$ denote its translation after applying the metric conversion factor. Under the Habitat camera convention, the goal-relative forward and left coordinates are

$$\Delta x_t^g = t_z, \qquad \Delta y_t^g = -t_x. \tag{40}$$

To recover relative yaw, the goal camera's forward axis $(0, 0, 1)^\mathsf{T}$ is transformed into the current camera frame:

$$\boldsymbol{f}_g = \boldsymbol{R}_{C_tC_g} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \qquad \Delta\theta_t^g = \mathrm{atan2}(-f_{g,x}, f_{g,z}). \tag{41}$$

The translation is then divided by the waypoint spacing before forming the relative-pose condition.

Habitat R-SCoRe settings. The input RGB observation is processed at an encoder size whose shorter side is 480 pixels. The Habitat camera intrinsics correspond to a 90° horizontal field of view; the focal length used by the pose solver is therefore set to half the image width, with the principal point at the image center.

The original R-SCoRe evaluation considers 10 retrieved candidates, whereas our Habitat implementation retrieves 20. Retrieval is implemented with product-quantized nearest-neighbor search over

the reference-image NetVLAD descriptors. For every retrieved candidate, its stored global scene encoding is concatenated with the local query descriptors before scene-coordinate prediction.

We use a maximum PnP–RANSAC reprojection error of 10 pixels and at most 10,000 RANSAC iterations. A separate pose is estimated for every retrieved candidate, and the candidate producing the largest inlier set is selected.

The R-SCoRe scene coordinates used by the Habitat localization map are converted to metres using a factor of 100. The waypoint spacing is

$$\ell = 0.2172\,\mathrm{m}. \tag{42}$$

The fixed dataset-level translation and yaw scales in the Habitat motion profile are

$$d_s = 0.95824505, \qquad \theta_s = 0.87605652. \tag{43}$$

Localization confidence. The RANSAC inlier count is converted into a bounded diagnostic confidence:

$$c_j = \mathrm{clip}\left(\frac{\left|\mathcal{I}_j^{(r^*)}\right| - 100}{1000 - 100}, 0, 1\right). \tag{44}$$

Thus, estimates with at most 100 inliers receive zero confidence, while estimates with at least 1000 inliers receive full confidence.

The confidence and inlier count are recorded for localization diagnostics. They are not additional obstacle-avoidance, safety, or navigation-control signals. The relative-pose conditioner described in the main paper receives the four-dimensional $(\Delta x, \Delta y, \sin\Delta\theta, \cos\Delta\theta)$ condition. The numerical localization settings above describe the Habitat interface and do not necessarily apply to the trajectory-prediction datasets or the Go Stanford experiment.

B Evaluation Protocol and Metric Definitions

Trajectory prediction. All methods are evaluated on the same fixed 100 test queries per dataset with an eight-step prediction horizon. Let $\hat{\boldsymbol{p}}_{n,k} \in \mathbb{R}^2$ and $\boldsymbol{p}^*_{n,k} \in \mathbb{R}^2$ denote the predicted and reference planar waypoint positions for query n at rollout step k, after converting all methods to a common waypoint coordinate system. Following the implementation used to produce the reported results, we compute a trajectory-level root-mean-square absolute trajectory error and a one-step relative pose error:

$$\begin{aligned}
\mathrm{ATE} &= \frac{1}{N}\sum_{n=1}^{N}\sqrt{\frac{1}{K}\sum_{k=1}^{K}\left\|\hat{\boldsymbol{p}}_{n,k} - \boldsymbol{p}^*_{n,k}\right\|_2^2}, \\
\mathrm{RPE} &= \frac{1}{N}\sum_{n=1}^{N}\sqrt{\frac{1}{K-1}\sum_{k=1}^{K-1}\left\|(\hat{\boldsymbol{p}}_{n,k+1} - \hat{\boldsymbol{p}}_{n,k}) - \left(\boldsymbol{p}^*_{n,k+1} - \boldsymbol{p}^*_{n,k}\right)\right\|_2^2},
\end{aligned} \tag{45}$$

where $K = 8$. ATE measures the absolute deviation of the complete predicted trajectory, while RPE measures the error in consecutive relative translations. Neither metric includes rotational or image-reprojection error. The four-dataset average is the unweighted mean over RECON, SACSoN, TartanDrive, and SCAND.

Closed-loop navigation. The Gibson evaluation uses 100 fixed validation episodes from the known Mosquito scene, a 2 m geodesic success threshold, and at most 100 executed actions. At each planning cycle, G^{2}WAM observes an RGB frame, updates its causal history, estimates goal-relative pose and motion, predicts eight latent/action steps, executes only the first waypoint through `VelocityXZYawMove`, and replans after the next observation. The evaluated controller does not decode future RGB frames or add CEM, APF, depth-based safety control, persistent bypass, or anti-spin logic. After each executed action, the evaluator uses geodesic distance to trigger an oracle STOP when the agent is within 2 m. No success check is performed at reset, so every episode executes at least one action. Geodesic distance is otherwise not provided as a control input.

Success rate (SR) is the fraction of successful episodes. Success weighted by path length (SPL) weights each success by the shortest-path length divided by the larger of that length and the executed path length. Distance to goal (DTG) is the mean final geodesic distance over all episodes, including failures. The model-only latency reported in the main text excludes localization, simulation, and action execution. The G^{2}WAM-GT comparison uses oracle pose and causal motion; the deployed G^{2}WAM setting uses image-estimated inputs.

C Supplementary Code and Navigation Videos

Code package. The anonymous supplementary archive provides the G^{2}WAM model definition, evaluation entry points for eight-step trajectory prediction and Gibson navigation, pose and causal-motion interfaces, fixed test-split metadata, configuration files, and validation scripts. It records checksums for included files and identifiers for the frozen assets used in our experiments. Raw datasets and third-party components, including DINOv3, R-SCoRe, and Habitat-Sim, are not redistributed. The training implementation and pretrained checkpoints are withheld during anonymous review and will be released after acceptance. Accordingly, the archive documents the evaluation pipeline but is not a self-contained, weight-inclusive reproduction.

Qualitative navigation videos. The ten paired videos show actual first-person RGB observations rendered by Habitat-Sim after executed actions, rather than images generated by either world model. Each pair uses the same Gibson Mosquito episode, starting observation, and goal image. For G^{2}WAM, we replay the original evaluation actions and verify the geodesic distance-to-goal trace at every step. NWM is freshly rerun using its six-step distilled checkpoint because repeated closed-loop inference with the original checkpoint is impractically slow. NWM is stochastic, so an individual recording need not match the outcome of an earlier aggregate evaluation. Playback is aligned by executed action index rather than wall-clock inference time; after one side terminates, its final frame remains visible.

Table 6: Outcome-selected paired Habitat recordings. S/F denotes success or failure under the 2 m threshold. Steps are executed actions, not video frames. These examples do not estimate full-test-set SR or SPL.

Episode index	Episode ID	G^{2}WAM	Steps	NWM	Steps
92	3353	S	9	S	75
17	3403	S	21	F	100
29	3532	S	60	S	36
19	3537	S	32	F	100
59	3310	S	12	S	1
16	3527	S	25	S	4
64	3408	S	20	F	100
24	3557	S	46	F	100
1	3595	S	26	F	100
83	3485	F	100	F	100

The recordings were deliberately selected for qualitative inspection: four pairs succeed for both methods, five succeed only for G^{2}WAM, and one fails for both. This outcome-conditioned subset is not representative of the complete 100-episode evaluation. Quantitative comparisons should therefore use the main evaluation table rather than these selected videos.

D AI Use Statement

Generative AI tools were used for language editing, literature search, drafting portions of the manuscript, software implementation, and assisting with the reproduction of baseline methods. All AI-assisted text, code, references, and baseline implementations were reviewed and verified by the authors. The authors take full responsibility for the final manuscript, methods, results, claims, and artifacts.