UNCERTAINTY-AWARE WORLD MODEL FOR AERIAL IMAGE-GOAL NAVIGATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Aerial image-goal navigation requires an unmanned aerial vehicle (UAV) to reach a target location specified by a goal image. Existing world-model-based methods rank candidate trajectories using predicted futures, but typically rely on only one or a few point predictions, which is inadequate for large-scale outdoor environments with substantial future-state uncertainty. To address this limitation, we propose the **U**ncertainty-**A**ware **N**avigation **W**orld **M**odel (**UA-NWM**), an efficient latent world model for aerial image-goal navigation, which formulates trajectory scoring as conditional out-of-distribution detection. UA-NWM represents plausible futures with an uncertainty subspace and decomposes the prediction–goal discrepancy into uncertainty-explainable and unexplainable components. Only the unexplainable residual is used for scoring, enabling robust selection without multiple future samples. Extensive experiments show that UA-NWM consistently outperforms existing navigation world models while maintaining low inference latency. Real-world UAV experiments further validate its practical applicability.

1 INTRODUCTION

Image-goal navigation maps current observation and a goal image to actions that guide an embodied agent toward the target location. Conventional methods (Sridhar et al., 2024; Shah et al., 2023b;a; Gode et al., 2025; Ren et al., 2025) directly predict actions or trajectories, whereas recent approaches (Bar et al., 2025; Zhang et al., 2025a; Yan et al., 2026; Zhang et al., 2026; 2025d; Shen et al., 2026) use world models to score candidate trajectories by predicting their future states, and select the one most likely to reach the goal for execution.

Existing world models can be broadly categorized as stochastic or deterministic. Stochastic world models (Bruce et al., 2024; Zhu et al., 2025; Lu et al., 2025; Agarwal et al., 2026) can generate different plausible futures, but are typically computationally expensive. Some studies (Zhou et al., 2025; Baldassarre et al., 2025; Karypidis et al., 2026; Chen et al., 2026a) therefore perform prediction in the latent spaces of vision foundation models (VFMs) (Oquab et al., 2024; Siméoni et al., 2025) to improve efficiency, but most of them remain deterministic, and can only produce a single estimate, rather than representing the full future-state distribution. Recent methods (Boduljak et al., 2025; Kerssies et al., 2026) begin exploring stochastic prediction in VFM latent spaces.

Despite these advances, a critical limitation remains—*future-state uncertainty* is not fully exploited in world-model-based trajectory scoring for navigation. This issue is especially pronounced in large-scale outdoor scenes, such as urban aerial navigation, where long-horizon motion and unseen regions yield multiple plausible futures under the same context and actions. As illustrated in Figure 1, occlusion by the building leaves the future appearance of the left region ambiguous. Under the manifold hypothesis (Bengio et al., 2013), these plausible futures concentrate around a low-dimensional manifold. Trajectory scoring can therefore be viewed as determining whether the goal image lies within the conditional future-state distribution.

However, existing navigation world models typically compare the goal with only one or a few point predictions, as shown in Figure 2. Deterministic models collapse multiple plausible futures into a single over-smoothed estimate that may even lie outside the distribution's high-density region, which is unreliable for scoring. Stochastic models can capture diverse plausible futures, but determining whether the goal belongs to this distribution requires sufficient samples to approximate it, incurring high computational cost. Relying on only one or a few samples makes scoring sensitive to sampling

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.

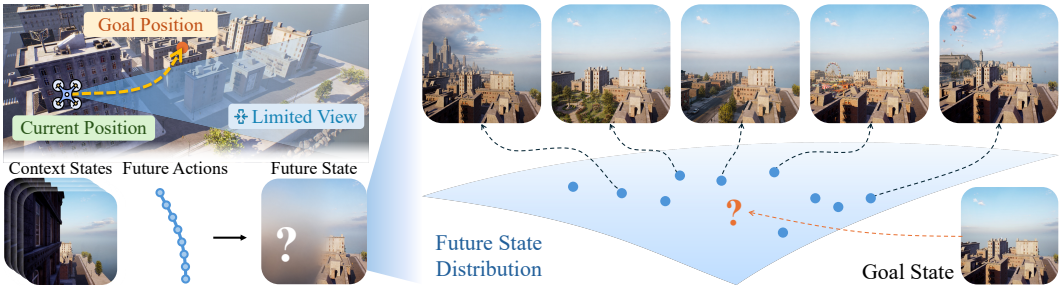


Figure 1: Illustration of *future-state uncertainty*.

randomness. Under *future-state uncertainty*, both paradigms are prone to assigning low scores to promising candidates and incorrectly rejecting them, thereby degrading navigation performance.

To address these limitations, we formulate trajectory scoring as a conditional out-of-distribution (OOD) detection problem. Each candidate trajectory defines a conditional future-state distribution, where the goal state is in-distribution if the trajectory can reach it and out-of-distribution otherwise. Since this distribution can only be modeled implicitly, directly determining whether the goal lies within it remains challenging. Instead of sampling multiple futures, we approximate this distribution with a predicted uncertainty subspace, as shown in Figure 2. Trajectories are scored by the discrepancy that remains unexplained by this subspace, avoiding reliance on either an over-smoothed deterministic prediction or an arbitrary stochastic sample.

Based on this formulation, we propose the **U**ncertainty-**A**ware **N**avigation **W**orld **M**odel (**UA-NWM**) for aerial image-goal navigation. UA-NWM uses a lightweight deterministic backbone to predict future representations in the DINOv3 (Siméoni et al., 2025) latent space, and introduces a Hierarchical Error Projection (HEP) module to model an uncertainty subspace around each prediction. HEP decomposes the discrepancy between the predicted and goal representations into uncertainty-explainable components and a remaining unexplained residual. Only the unexplained part is used for trajectory scoring, avoiding penalties on plausible uncertainty-induced deviations.

We construct a high-quality dataset from existing aerial navigation benchmarks and train UA-NWM with a staged optimization scheme. Extensive offline and online experiments demonstrate that UA-NWM substantially outperforms existing methods while maintaining low inference latency, with the uncertainty-aware HEP consistently improving trajectory-scoring accuracy. Real-world UAV experiments further validate its effectiveness for real application.

Our main contributions are summarized as follows:

- We propose UA-NWM, a navigation world model that formulates trajectory scoring as conditional OOD detection and represents plausible future states with an uncertainty subspace.
- We introduce a Hierarchical Error Projection (HEP) module that decomposes the prediction–goal discrepancy into uncertainty-explainable and unexplainable components, using only the latter for robust trajectory scoring.
- UA-NWM achieves state-of-the-art performance in both offline and online evaluations while maintaining low inference latency. Its effectiveness and practicality are further demonstrated through real-world UAV experiments.

2 Related Work

2.1 Image-Goal Navigation

Image-goal navigation guides an agent to an image-specified position, supporting applications such as delivery, inspection, and rescue when precise coordinates are unavailable.

Classical methods rely on mapping or localization for collision-free planning (Bonin-Font et al., 2008; Zhang et al., 2022; Wu et al., 2022; Guo et al., 2025), but may accumulate errors in com-

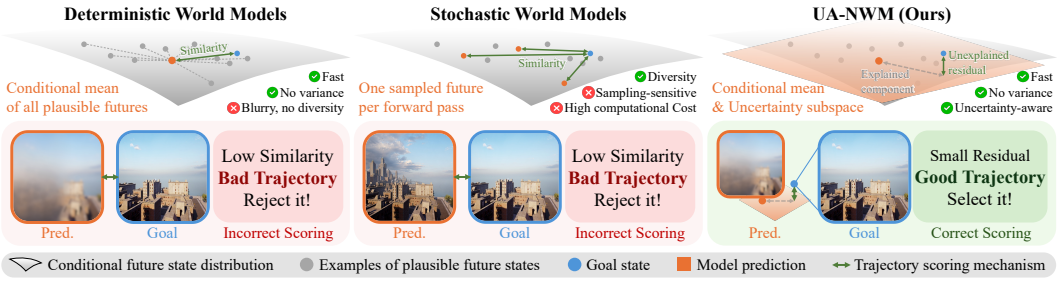


Figure 2: Comparison of UA-NWM and previous navigation world models in terms of future prediction and trajectory scoring.

plex scenes. Learning-based approaches directly predict from observations and goals. Early methods (Zhu et al., 2017) use reinforcement learning, while recent studies (Shah et al., 2023a;b; Yin et al., 2025) improve generalization with heterogeneous robot data, Transformer policies, or LLM backbones. Generative policies (Sridhar et al., 2024; Gode et al., 2025; Zeng et al., 2025; Ren et al., 2025; Chen et al., 2026b) further model multimodal action or trajectory distributions.

Recent work has extended image-goal navigation to UAVs, but they mainly target indoor exploration (Yan et al., 2025; Gao et al., 2026b), language guidance (Zhang et al., 2025b), or local pose alignment (Li et al., 2026). These methods generally lack action-conditioned prediction of multi-step visual outcomes for candidate trajectories, limiting reliable long-horizon planning in complex outdoor environments.

2.2 World Models for Navigation

World models have been widely adopted in embodied tasks, while the broader idea of future prediction has also been explored across diverse visual applications (Ma et al., 2026; Zhu et al., 2026). For navigation, world models allow agents to score candidate actions by predicting their future visual consequences. Early work (Koh et al., 2021; Wang et al., 2023; 2025) explored future-scene imagination for indoor and long-horizon navigation, while NWM (Bar et al., 2025) introduced action-conditioned video prediction for trajectory scoring. Following NWM, recent methods have improved prediction efficiency and representation quality. One-Step WM (Shen et al., 2026) generates future observations in a single step, while LS-NWM (Zhang et al., 2025c), ReL-NWM (Zhang et al., 2025d), and RAE-NWM (Zhang et al., 2026) perform prediction in dense VFM feature spaces. Another line of work (Dong et al., 2025; Mei et al., 2026; Azuma et al., 2026) jointly integrates future prediction with trajectory planning. In aerial navigation, WorldVLN (Zhao et al., 2026) and ImagineUAV (Liu et al., 2026) focus on language-conditioned flight, while ANWM (Zhang et al., 2025a) uses depth-assisted future-frame projection for image-goal navigation.

Despite these advances, existing methods typically score each trajectory using only one or a few predicted futures. Such point-wise comparison cannot distinguish plausible future variations from action-inconsistent errors. Deterministic models collapse multiple outcomes into a single blurry estimate, while stochastic models remain sensitive to sampling variance. This motivates an explicit representation of the future-state distribution for trajectory scoring.

3 Preliminaries

In image-goal navigation, a world model can serve as a trajectory ranker rather than directly predicting the next control command. At time t, the agent observes a context of C images, $O_{t-C+1:t} = \{o_{t-C+1}, \dots, o_t\}$, and scores action sequences over a horizon of H steps. Each candidate sequence is denoted by $A = \{a_t, \dots, a_{t+H-1}\}$. A proposal policy or sampling-based planner generates a finite candidate set $\mathcal{A} = \{A^{(1)}, \dots, A^{(Q)}\}$. For each candidate trajectory, the world model predicts its terminal visual representation and evaluates its compatibility with the goal image o_g. Formally, the world model assigns each candidate a trajectory cost

$$s(A, o_g \mid O_{t-C+1:t}) = D\left(F_\theta(O_{t-C+1:t}, A), \phi(o_g)\right), \tag{1}$$

where $\phi(\cdot)$ is a frozen visual feature extractor, F_θ predicts the future feature representation conditioned on the observation context and candidate actions, and $D(\cdot,\cdot)$ measures their discrepancy. The optimal action sequence is then selected as

$$A^\star = \mathrm{argmin}_{A\in\mathcal{A}} s(A, o_g \mid O_{t-C+1:t}). \tag{2}$$

Existing world-model rankers typically define D as a point-to-point distance. Pixel-space models compare the generated future image with the goal using perceptual metrics such as LPIPS (Zhang et al., 2018), whereas latent-space models measure the cosine or Euclidean distance between feature representations. This formulation also applies to standalone planning, where $\mathcal{A}$ is iteratively refined by a sampling-based optimizer such as the cross-entropy method (CEM) (Rubinstein, 1997), with the cost s serving as the optimization objective.

4 METHOD

UA-NWM scores candidate trajectories by measuring goal compatibility with the future-state distribution conditioned on the current context and each candidate action sequence. We first formulate the idea of uncertainty-aware trajectory scoring and introduce the Hierarchical Error Projection (HEP) module, then describe the deterministic latent world model backbone and training procedure.

4.1 UNCERTAINTY-AWARE TRAJECTORY SCORING

Given context observations $O_{t-C+1:t}$ and a candidate action sequence $A_{t:t+H-1}$, the latent world model makes a deterministic prediction of the final-horizon DINO feature map:

$$\mu = F_\theta(O_{t-C+1:t}, A_{t:t+H-1}), \tag{3}$$

where μ contains N patch tokens. Let $x_g = \phi(o_g)$ denote the DINO feature map of the goal o_g. Their normalized discrepancy is

$$e = \mathrm{norm}(x_g) - \mathrm{norm}(\mu), \tag{4}$$

where $\mathrm{norm}(\cdot)$ denotes channel-wise ℓ_2 normalization.

Previous deterministic scorers penalize the full discrepancy e. We instead formulate trajectory scoring as conditional OOD detection. For each candidate trajectory, its future-state distribution defines an in-distribution region in the DINO feature space. As shown in Figure 3(a), we approximate this region with an uncertainty subspace $\mathcal{S}$ around μ and decompose e into an explainable component $e^{\parallel}$ and a remaining residual $e^{\perp}$[1]. Only the unexplained residual $e^{\perp}$ is penalized:

$$s(A_{t:t+H-1}, o_g \mid O_{t-C+1:t}) = \frac{1}{N}\sum_{i=1}^{N} \left\| e_i^{\perp} \right\|_2^2, \tag{5}$$

where $e_i^{\perp}$ is the residual at the i-th patch of $e^{\perp}$.

Therefore, a candidate receives a low cost when the goal differs from the predicted mean primarily along plausible uncertainty directions, and a high cost when the discrepancy cannot be explained by $\mathcal{S}$. In Figure 3(b), trajectories A and B have similar total discrepancies e. For trajectory A, most of the discrepancy between $\mu^{(A)}$ and x_g can be explained by occlusion, yielding a small $e^{\perp}$. For trajectory B, however, the tree and road in the lower-right region of x_g is inconsistent with the plausible future appearances around $\mu^{(B)}$, where this region is expected to contain buildings, resulting in a large $e^{\perp}$. Our score thus correctly favors trajectory A.

4.2 HIERARCHICAL ERROR PROJECTION (HEP)

To decompose e into $e^{\parallel}$ and $e^{\perp}$, we introduce the Hierarchical Error Projection (HEP) module, illustrated in Figure 3(c). HEP takes the deterministic prediction μ and context observation latents

[1]The superscripts $\parallel$ and $\perp$ are intuitive notation and do not imply exact orthogonality to a single global subspace. The precise computation is detailed in Section A.7.2

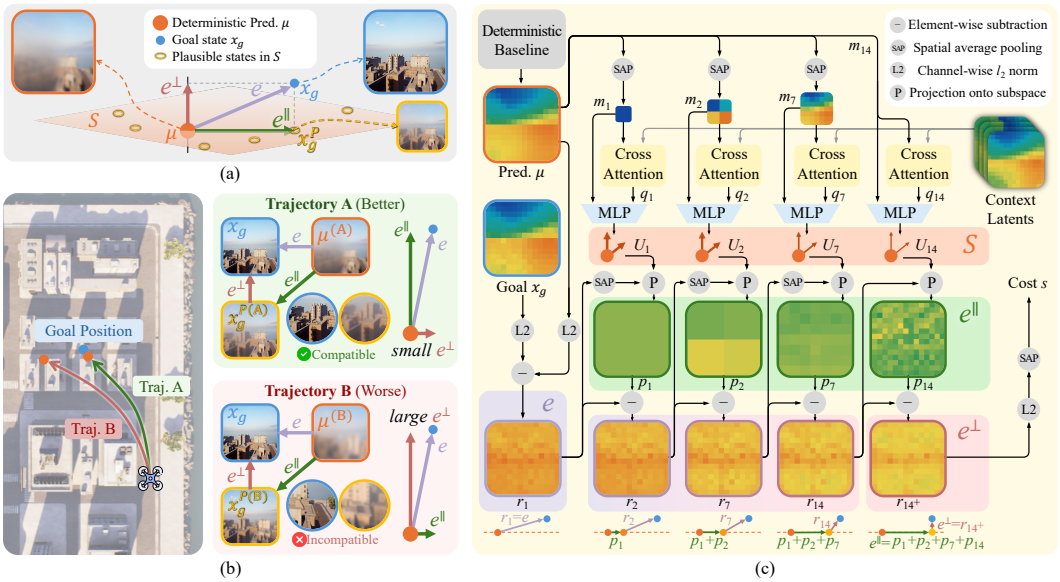


Figure 3: (a) Illustration of the uncertainty-aware trajectory scoring idea. Conventional deterministic methods score based on e. We divide e as $e^\perp$ and $e^\parallel$, and use only $e^\perp$ for scoring. (b) An example of how uncertainty-aware trajectory scoring can improve navigation performance. (c) The overall pipeline of our proposed Hierarchical Error Projection (HEP) module.

as input and models plausible deviations using multi-scale low-rank subspaces. To capture spatially coherent uncertainty, it decomposes the discrepancy field through a coarse-to-fine grid pyramid

$$\mathcal{G} = \{1, 2, 7, 14\}. \tag{6}$$

At each scale g, spatial average pooling partitions the 14×14 patch grid of μ into $g \times g$ cells and yields a descriptor $m_{g,c}$ for each cell c. Then a context descriptor $q_{g,c}$ is obtained through cross-attention with context latents, enabling each cell to retrieve relevant contextual evidence. Finally, a scale-specific MLP f_g generates a rank-R basis for each cell:

$$U_{g,c} = f_g([m_{g,c}, q_{g,c}]) = \{u_{g,c,1}, \ldots, u_{g,c,R}\}, \quad u_{g,c,r} \in \mathbf{R}^D, \tag{7}$$

where $[\cdot, \cdot]$ denotes concatenation.

HEP projects the discrepancy hierarchically from coarse to fine. It maintains a residual field r containing the discrepancy unexplained by coarser scales. Let $r_{g,i} \in \mathbb{R}^D$ denote the residual at patch i upon entering scale g, initialized as $r_{1,i} = e_i$. At each scale, the residuals within each cell c are averaged:

$$\bar{r}_{g,c} = \frac{1}{|\Omega_{g,c}|} \sum_{i \in \Omega_{g,c}} r_{g,i}, \tag{8}$$

where $\Omega_{g,c}$ denotes the set of patches in cell c. We then project $\bar{r}_{g,c}$ onto the subspace spanned by $U_{g,c}$ by solving a ridge-regularized least-squares problem:

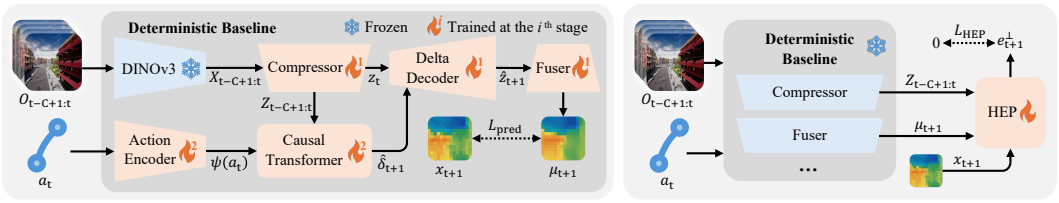
$$\alpha_{g,c} = \underset{\alpha \in \mathbf{R}^R}{\operatorname{argmin}} \left\| \bar{r}_{g,c} - \sum_{r=1}^{R} \alpha_r u_{g,c,r} \right\|_2^2 + \lambda \|\alpha\|_2^2, \tag{9}$$

where λ is a small regularization coefficient. The resulting uncertainty-explainable component is

$$p_{g,c} = \sum_{r=1}^{R} \alpha_{g,c,r} u_{g,c,r}, \tag{10}$$

which is subtracted from every patch in the cell to obtain the residual for the next finer scale:

$$r_{g^+,i} = r_{g,i} - p_{g,c}, \quad i \in \Omega_{g,c}, \tag{11}$$



(a) Deterministic baseline training (b) HEP training

Figure 4: The overall training procedure of UA-NWM.

where g^+ denotes the next finer scale in $\mathcal{G}$. After processing the finest scale, the remaining residual defines $e^\perp$, while the explained component is given by $e^\parallel = e - e^\perp$. Overall, HEP constructs a coarse-to-fine, region-specific uncertainty subspace that approximates the conditional future-state distribution around the deterministic prediction. This enables uncertainty-aware trajectory scoring in one forward pass without sampling multiple future observations.

4.3 Deterministic Baseline and Training

UA-NWM builds on a deterministic latent world model. A frozen DINOv3 encoder extracts dense features $x_t = \phi(o_t)$, which are compressed into K latent tokens $z_t = C_\eta(x_t)$. Following Delta-World (Kerssies et al., 2026), the transition from z_t to z_{t+1} is represented by M delta tokens encoding their differences. A causal Transformer predicts these delta tokens rather than the next state directly:

$$\hat{\delta}_{t+1} = P_\omega(Z_{t-C+1:t}, \psi(a_t)), \tag{12}$$

$$\hat{z}_{t+1} = D_\Delta(z_t, \hat{\delta}_{t+1}), \tag{13}$$

$$\mu_{t+1} = G_\zeta(\hat{z}_{t+1}), \tag{14}$$

where D_Δ is the delta decoder and G_ζ fuses the predicted tokens into the DINO feature map μ_{t+1}.

The deterministic baseline is trained in two stages. First, the compressor, fuser, and delta decoder are optimized to learn compact representations and latent transitions. These modules are then frozen, while the action encoder and causal Transformer are trained to predict future latent states. The prediction μ_{t+1} is directly supervised by the DINO feature of the future observation $x_{t+1} = \phi(o_{t+1})$. Further training details are provided in Section A.2.2.

With the deterministic baseline frozen, HEP is then trained to minimize the fraction of $e^\perp$:

$$\mathcal{L}_{\text{HEP}} = \frac{\sum_{i=1}^{N} \|e^\perp_{t+1,i}\|_2^2}{\sum_{i=1}^{N} \|e_{t+1,i}\|_2^2 + \epsilon}. \tag{15}$$

The normalized objective prevents large-error samples from dominating training, whereas inference uses the unnormalized residual energy for trajectory scoring. The prediction and HEP losses are averaged over an eight-step autoregressive rollout to mitigate error accumulation.

5 Experiments

5.1 Experimental Settings

5.1.1 Benchmark

Existing UAV navigation benchmarks mainly target vision-and-language navigation (VLN) or indoor environments, leaving no public benchmark for large-scale outdoor 3D UAV image-goal navigation available. Therefore, we construct a new benchmark named **AirGoal-10k**, using the AirSim simulator (Shah et al., 2017) and building upon existing aerial VLN benchmarks (Liu et al., 2023; Gao et al., 2025). AirGoal-10k contains 9000 trajectories for training, 1000 trajectories for validation, and 1000 trajectories for testing. Each trajectory comprises 3D waypoints paired with egocentric RGB observations. Further details are provided in Section A.3.

Table 1: Comparison with world-model-based methods on trajectory ranking task of AirGoal-10k.

Methods	#Params	8 Candidates		16 Candidates		32 Candidates		Time/Frame ↓
		ATE ↓	RPE ↓	ATE ↓	RPE ↓	ATE ↓	RPE ↓	
Random Selection	–	1.76	0.48	1.79	0.49	1.80	0.49	–
NWM	195M	1.47	0.40	1.40	0.38	1.37	0.37	438.67 ms
One-Step WM	176M	1.52	0.41	1.48	0.40	1.44	0.39	10.44 ms
MWM	202M	1.58	0.43	1.60	0.43	1.59	0.43	122.78 ms
RAE-NWM	223M	1.54	0.42	1.54	0.42	1.50	0.41	386.00 ms
Det. Baseline (Ours)	115M	1.39	0.38	1.33	0.36	1.27	0.35	8.06 ms
UA-NWM (Ours)	122M	**1.26**	**0.35**	**1.17**	**0.33**	**1.09**	**0.30**	8.47 ms
Oracle Selection	–	0.90	0.26	0.76	0.22	0.63	0.19	–

Table 2: Comparison with policy-based and world-model-based methods on standalone planning task of AirGoal-10k.

Methods	ATE ↓	RPE ↓	VENUE
Policy-based Methods			
GNM	1.29	0.35	ICRA'2023
ViNT	1.37	0.38	CoRL'2023
NoMaD	1.79	0.49	ICRA'2024
FlowNav	1.81	0.50	IROS'2025
NaviBridger$_{\text{CVAE}}$	1.58	0.44	CVPR'2025
World-model-based Methods			
NWM	1.40	0.39	CVPR'2025
One-Step WM	1.53	0.44	Arxiv'2026
MWM	1.54	0.43	Arxiv'2026
RAE-NWM	1.46	0.43	ECCV'2026
Det. Baseline (Ours)	1.26	0.35	–
UA-NWM (Ours)	**1.22**	**0.33**	–

Table 3: Comparison with policy-based and world-model-based methods on online closed-loop navigation task.

Methods	SR ↑	SPL ↑	Time/Step ↓
Policy-based Methods			
GNM	48.0%	40.9%	37.46 ms
ViNT	53.0%	46.0%	48.47 ms
NoMaD	56.0%	47.2%	117.49 ms
FlowNav	45.0%	38.6%	117.55 ms
NaviBridger$_{\text{CVAE}}$	48.0%	41.8%	166.08 ms
World-model-based Methods			
NWM	63.0%	52.1%	191.71 s
One-Step WM	55.0%	43.5%	3.39 s
MWM	69.0%	56.0%	66.57 s
RAE-NWM	70.0%	57.8%	202.54 s
Det. Baseline (Ours)	70.0%	60.1%	2.38 s
UA-NWM (Ours)	**76.0%**	**64.5%**	2.70 s

5.1.2 Evaluation Metrics

For offline evaluation, we use Absolute Trajectory Error (ATE) and Relative Pose Error (RPE), following Bar et al. (2025). For online closed-loop evaluation, we report Success Rate (SR), Success weighted by Path Length (SPL), and the average planning time per step, following Zhang et al. (2025d). Success requires stopping within 20 meters of the target location within the step budget.

5.1.3 Implementation Details

We use a frozen DINOv3 ViT-B/16 (Siméoni et al., 2025) to extract dense visual features. Unless otherwise specified, the context length is $C = 4$, the autoregressive prediction horizon is $H = 8$, and each visual state is compressed into $K = 32$ latent tokens. The model is trained with AdamW on four NVIDIA RTX 4090 GPUs. Further details are provided in Section A.2.

5.2 Offline Experiments

We evaluate two offline settings: trajectory ranking with externally generated candidates and standalone planning without an external policy. For trajectory ranking, NoMaD (Sridhar et al., 2024) generates 8, 16, or 32 candidate action sequences, which are then scored and selected by each world-model-based method using identical candidate sets for fairness. As shown in Table 1, UA-NWM achieves the lowest ATE and RPE across all candidate-set sizes while remaining substantially faster

Table 4: Ablation on HEP's scale components.

Scales g				16 Candidates		32 Candidates	
1	2	7	14	ATE ↓	RPE ↓	ATE ↓	RPE ↓
×	×	×	✓	1.22	0.34	1.15	0.32
×	✓	✓	✓	1.20	**0.33**	1.12	0.31
✓	×	✓	✓	1.21	0.34	1.13	0.31
✓	✓	×	✓	1.21	0.34	1.14	0.32
✓	✓	✓	×	1.24	0.34	1.17	0.32
✓	✓	✓	✓	**1.17**	**0.33**	**1.09**	**0.30**

Table 5: Effect of the basis rank R.

Rank R	16 Candidates		32 Candidates	
	ATE ↓	RPE ↓	ATE ↓	RPE ↓
1	1.23	0.34	1.16	0.32
2	**1.17**	**0.33**	**1.09**	**0.30**
3	1.20	**0.33**	1.13	0.31
4	1.21	0.34	1.14	0.32
5	1.21	**0.33**	1.14	0.32

than previous methods, especially stochastic models requiring iterative generation. For standalone planning, policy-based methods directly predict one trajectory, whereas world models optimize action sequences with CEM (Rubinstein, 1997) following NWM (Bar et al., 2025). At each of three iterations, CEM samples and scores 32 candidates and updates the sampling distribution using the top 16 elites, after which the final distribution mean is used as the planned trajectory. As shown in Table 2, UA-NWM still achieves the best planning performance.

5.3 Online Simulation Experiments

We evaluate closed-loop navigation on 100 AirSim episodes. In each episode, the agent navigates to an unknown-distance goal location specified by an image. At each step, policy-based methods directly predict a trajectory, whereas world models plan with CEM. As shown in Table 3, UA-NWM achieves the best online performance. Compared with offline planning, this setting introduces a distribution shift from fixed-length trajectories to variable-length, long-horizon navigation. Policy-based methods degrade substantially, while autoregressive world models generalize more robustly.

5.4 Ablation Studies

Hierarchical structure. We ablate each scale of HEP to assess its contribution. As shown in Table 4, removing any scale degrades performance. Scale 14 contributes the most, suggesting that explainable errors $e^{\parallel}$ predominantly arise from fine-grained local variations, whereas Scale 1 contributes the least because such variations rarely span the entire image.

Rank of the basis. We vary the basis rank R across all scales. As shown in Table 5, $R = 2$ performs best, suggesting that a low-dimensional subspace is sufficient to capture the dominant directions of $e^{\parallel}$. A smaller rank underrepresents future uncertainty, whereas a larger rank may absorb unexplainable errors and weaken trajectory discrimination.

5.5 Qualitative Analysis

Figure 5 compares predictions under ground-truth trajectories. NWM and RAE-NWM exhibit substantial error accumulation during multi-step autoregressive rollout. For UA-NWM, the deterministic backbone predicts a blurry mean future latent μ, which may deviate considerably from x_g when multiple futures are plausible. HEP decomposes their residual e. The reconstructed state $x_g^P = \mu + e^{\parallel} = x_g - e^{\perp}$ incorporates the explainable components. Its remaining discrepancy from x_g is exactly $e^{\perp}$, which is substantially smaller than e and is used for trajectory scoring.

Sampling additional points from $\mathcal{S}$ yields various plausible futures, demonstrating that it captures meaningful variations beyond a single deterministic prediction, which further reveal two major sources of *future-state uncertainty* in navigation:

- **Occlusion-induced ambiguity.** In the first example, foreground trees limit the initial observation, leaving both tree coverage and roof structure in future observation uncertain. The goal image represents one plausible future with no trees in view and a curvilinear gable on the roof. While μ averages over these possibilities, x_g^P recovers this goal-consistent state.

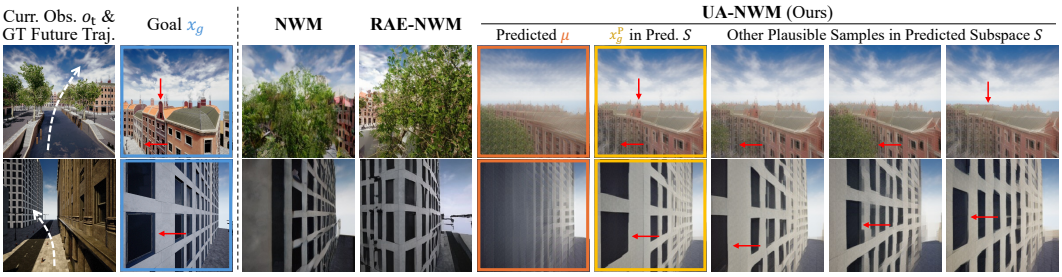


Figure 5: Qualitative comparison of future predictions. The DINO latent features predicted by RAE-NWM and UA-NWM are decoded using RAE (Zheng et al., 2025) and RAEv2 (Singh et al., 2026), respectively, solely for visualization. RAE-NWM and UA-NWM perform trajectory scoring directly in the latent space, whereas NWM scores LPIPS (Zhang et al., 2018) of predicted RGB images. Details for latent visualization are provided in Section A.4.5.

Other samples show varying tree coverage or a roof without the gable, all plausible given the limited initial evidence.

- **Long-horizon drift.** Small pose errors accumulate over long rollouts, making fine-grained details uncertain even without occlusion. In the second example, the overall building layout remains predictable, while window positions vary substantially with minor drifts. Modeling this uncertainty too broadly may absorb true trajectory errors and weaken scoring, as seen in NWM and RAE-NWM. UA-NWM instead preserves global geometry while modeling plausible local variations, balancing uncertainty tolerance and trajectory discrimination.

5.6 Real-world Experiments

To evaluate real-world applicability, we deploy UA-NWM locally on a MacBook Air and test it on a custom-built quadrotor, as shown in Figure 6. The MacBook communicates with the UAV via a mobile hotspot, performs onboard planning, and transmits the predicted 3D waypoints to the low-level flight controller for execution. At each step, UA-NWM plans an 8-waypoint trajectory using CEM, executes the first waypoint, and replans from the latest observation. On-device planning takes approximately 9 seconds per trajectory, including communication latency. Experiments on five navigation tasks across multiple scenes demonstrate successful closed-loop deployment and zero-shot sim-to-real generalization without fine-tuning on real-world data. Details are provided in Section A.4.4.

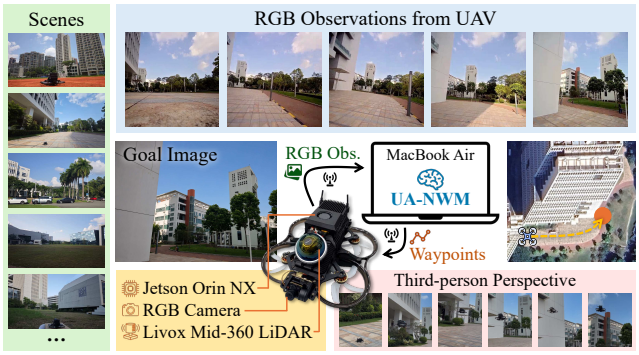


Figure 6: Illustration of real-world UAV experiments.

6 Conclusion

We presented UA-NWM, an uncertainty-aware world model for aerial image-goal navigation. UA-NWM formulates trajectory scoring as conditional OOD detection, models plausible future variations through an uncertainty subspace, and sequentially separates plausible uncertainty-induced deviations from unexplained residual errors. This enables efficient distribution-aware trajectory scoring without stochastic future sampling. Experiments show that UA-NWM improves navigation performance across diverse tasks, while preserving low inference latency, and real-world UAV deployment further supports its practical applicability.

AI USE STATEMENT

In this work, we used generative AI tools for assisting in the writing of proofs, implementing methods, assisting with translation, cleaning and reformatting dataset. We have not used generative AI tools for helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, supporting qualitative and thematic data analysis, or interpreting results; and generating synthetic data sets is not applicable to this work. Additionally, we used generative AI tools for creating or editing software code, sourcing/searching for information, and identifying relevant literature. We have reviewed all AI-assisted work. AI-assisted proof text and mathematical derivations were checked step by step by 2 authors. AI-generated or AI-modified code was manually reviewed and tested for correctness through relevant unit tests, smoke tests, and comparisons with expected outputs by 2 authors. Translated or AI-edited text was checked against the intended meaning and manually revised. The outputs of AI-assisted dataset cleaning and reformatting were validated using consistency checks and manual inspection of representative samples by 2 authors. Information and potentially relevant literature identified with AI assistance were verified against the original sources, and citations were included only after the authors inspected the corresponding works. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Anonymous source code, configuration files, and scripts for training, evaluation, visualization, and real-world deployment are available at `https://anonymous.4open.science/r/UA-NWM_Anonymous-852E`. The main paper specifies the model architecture, objectives, and evaluation protocols, while the appendix provides further details on the training procedure, dataset construction, baseline implementation, and experimental settings.

REFERENCES

Niket Agarwal, Arslan Ali, Jon Allen, Martin Antolini, Adeline Aubame, Alisson Azzolini, Junjie Bai, Maciej Bala, Yogesh Balaji, Josh Bapst, et al. Cosmos 3: Omnimodal world models for physical ai. *arXiv preprint arXiv:2606.02800*, 2026.

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.

Utkarsh Bajpai, Julius Rückin, Cyrill Stachniss, and Marija Popović. Uncertainty-informed active perception for open vocabulary object goal navigation. In *2025 European Conference on Mobile Robots (ECMR)*, pp. 1–7. IEEE, 2025.

Federico Baldassarre, Marc Szafraniec, Basile Terver, Vasil Khalidov, Francisco Massa, Yann LeCun, Patrick Labatut, Maximilian Seitzer, and Piotr Bojanowski. Back to the features: Dino as a foundation for video world models. *arXiv preprint arXiv:2507.19468*, 2025.

Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15791–15801. IEEE, 2025.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Gabrijel Boduljak, Yushi Lan, Christian Rupprecht, and Andrea Vedaldi. Vfmf: World modeling by forecasting vision foundation model features. *arXiv preprint arXiv:2512.11225*, 2025.

Francisco Bonin-Font, Alberto Ortiz, and Gabriel Oliver. Visual navigation for mobile robots: A survey. *Journal of Intelligent and Robotic Systems*, 53(3):263–296, 2008.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *International Conference on Machine Learning*, pp. 4603–4623. PMLR, 2024.

Jialei Chen, Kai Wang, Kang Chen, Shuaihang Chen, Feng Gao, Wenhao Tang, Zhiyuan Li, Weilin Liu, Zhuyu Yao, Boxun Li, et al. Lawam: Latent world action models for efficient dynamics-aware robot policies. *arXiv preprint arXiv:2606.15768*, 2026a.

Yuqi Chen, Junjie Gao, Yongzhou Pan, Siyuan Song, Zixuan Zhang, Jiaping Xiao, and Mir Feroskhan. Geninav: Generative model driven image-goal navigation via imagination-guided consistency flow matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 996–1005, 2026b.

Gabriele Costante, Christian Forster, Jeffrey Delmerico, Paolo Valigi, and Davide Scaramuzza. Perception-aware path planning. *arXiv preprint arXiv:1605.04151*, 2016.

Xin Ding, Jianyu Wei, Yifan Yang, Shiqi Jiang, Qianxi Zhang, Hao Wu, Fucheng Jia, Liang Mi, Yuxuan Yan, Weijun Wang, et al. Adanav: Adaptive reasoning with uncertainty for vision-language navigation. *arXiv preprint arXiv:2509.24387*, 2025.

Yifei Dong, Fengyi Wu, Guangyu Chen, Zhi-Qi Cheng, Qiyu Hu, Yuxuan Zhou, Jingdong Sun, Jun-Yan He, Qi Dai, and Alexander G Hauptmann. Unified world models: Memory-augmented planning and foresight for visual navigation. *arXiv preprint arXiv:2510.08713*, 2025.

Jianzhe Gao, Rui Liu, Yuxuan Xu, Tongtong Cao, Yingxue Zhang, Zhanguang Zhang, Sida Peng, Yi Yang, and Wenguan Wang. Uncertainty-aware gaussian map for vision-language navigation. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 126558–126581, 2026a.

Junjie Gao, Yuqi Chen, Yongzhou Pan, Yaosheng Deng, Jiaping Xiao, and Mir Feroskhan. Flying to image-specified objects: 3d quadrotor navigation via cross-graph memory and viewpoint planning. *arXiv preprint arXiv:2606.29917*, 2026b.

Yunpeng Gao, Chenhui Li, Zhongrui You, Junli Liu, Zhen Li, Pengan Chen, Qizhi Chen, Zhonghan Tang, Liansheng Wang, Penghui Yang, et al. Openfly: A comprehensive platform for aerial vision-language navigation. *arXiv preprint arXiv:2502.18041*, 2025.

Samiran Gode, Abhijeet Nayak, Débora NP Oliveira, Michael Krawez, Cordelia Schmid, and Wolfram Burgard. Flownav: Combining flow matching and depth priors for efficient navigation. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 17762–17768. IEEE, 2025.

Wenxuan Guo, Xiuwei Xu, Hang Yin, Ziwei Wang, Jianjiang Feng, Jie Zhou, and Jiwen Lu. Igl-nav: Incremental 3d gaussian localization for image-goal navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6808–6817, 2025.

Sudarshan S Harithas, Ayyappa Swamy Thatavarthy, Gurkirat Singh, Arun K Singh, and K Madhava Krishna. Urbanfly: Uncertainty-aware planning for navigation amongst high-rises with monocular visual-inertial slam maps. In *2023 American Control Conference (ACC)*, pp. 557–563. IEEE, 2023.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

Noriaki Hirose, Amir Sadeghian, Marynel Vázquez, Patrick Goebel, and Silvio Savarese. Gonet: A semi-supervised deep learning approach for traversability estimation. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 3044–3051. IEEE, 2018.

Ifueko Igbinedion and Sertac Karaman. Learning when to ask for help: Efficient interactive navigation via implicit uncertainty estimation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9593–9599. IEEE, 2024.

Efstathios Karypidis, Ioannis Kakogeorgiou, Spyridon Gidaris, and Nikos Komodakis. Dino-foresight: Looking into the future with dino. *Advances in Neural Information Processing Systems*, 38:163779–163811, 2026.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017.

Tommie Kerssies, Gabriele Berton, Ju He, Qihang Yu, Wufei Ma, Daan de Geus, Gijs Dubbelman, and Liang-Chieh Chen. A frame is worth one token: Efficient generative world modeling with delta tokens. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27978–27988, June 2026.

Jing Yu Koh, Honglak Lee, Yinfei Yang, Jason Baldridge, and Peter Anderson. Pathdreamer: A world model for indoor navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14738–14748, 2021.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018.

Yaxuan Li, Jiarui Zeng, Shaofei Huang, and Zhedong Zheng. Last-meter precision navigation for uavs: A diffusion-refined aerial visual servoing approach. *arXiv preprint arXiv:2607.04352*, 2026.

Shubo Liu, Hongsheng Zhang, Yuankai Qi, Peng Wang, Yanning Zhang, and Qi Wu. Aerialvln: Vision-and-language navigation for uavs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15384–15394, 2023.

Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection. *Advances in neural information processing systems*, 33:21464–21475, 2020.

Xuchen Liu, Jiawei Huang, Shihao Xia, Bingxi Liu, Jinqiang Cui, and Jiankun Yang. Imagineuav: Aerial vision-language navigation via world-action modeling and kinodynamic planning. *arXiv preprint arXiv:2606.01205*, 2026.

Guanxing Lu, Baoxiong Jia, Puhao Li, Yixin Chen, Ziwei Wang, Yansong Tang, and Siyuan Huang. Gwm: Towards scalable gaussian world models for robotic manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9263–9274, 2025.

Shilin Ma, Chubin Zhang, Changyuan Wang, Yuji Wang, Yue Wu, Zixuan Wang, Jingqi Tian, Zheng Zhu, and Yansong Tang. Safe-pruner: Semantic attention-guided future-aware token pruning for efficient vision-language-action manipulation. *arXiv preprint arXiv:2605.29662*, 2026.

Yanghong Mei, Longteng Guo, Ming-Ming Yu, Guiyu Zhao, Xingjian He, and Jing Liu. Navwm: A unified navigation world model for foresight-driven planning. *arXiv preprint arXiv:2606.24101*, 2026.

Zhiting Mei, Tenny Yin, Micah Baker, Ola Shorinwa, and Anirudha Majumdar. World models that know when they don't know: Controllable video generation with calibrated uncertainty. *arXiv preprint arXiv:2512.05927*, 2025.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? *arXiv preprint arXiv:1810.09136*, 2018.

Huan Nguyen, Rasmus Andersen, Evangelos Boukas, and Kostas Alexis. Uncertainty-aware visually-attentive navigation using deep neural networks. *The International Journal of Robotics Research*, 43(6):840–872, 2024.

David A Nix and Andreas S Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*, volume 1, pp. 55–60. IEEE, 1994.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research Journal*, 2024.

Yiyuan Pan, Yunzhe Xu, Zhe Liu, and Hesheng Wang. Seeing through uncertainty: Robust task-oriented optimization in visual navigation. *Advances in Neural Information Processing Systems*, 38:25259–25286, 2026.

Sara Pohland and Claire Tomlin. Competency-aware planning for probabilistically safe navigation under perception uncertainty. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 15291–15298. IEEE, 2025.

Sai Prasanna, Daniel Honerkamp, Kshitij Sirohi, Tim Welschehold, Wolfram Burgard, and Abhinav Valada. Perception matters: Enhancing embodied ai with uncertainty-aware semantic segmentation. *arXiv preprint arXiv:2408.02297*, 2024.

Hao Ren, Yiming Zeng, Zetong Bi, Zhaoliang Wan, Junlong Huang, and Hui Cheng. Prior does matter: Visual navigation via denoising diffusion bridge models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12100–12110, 2025.

Jie Ren, Peter J Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark Depristo, Joshua Dillon, and Balaji Lakshminarayanan. Likelihood ratios for out-of-distribution detection. *Advances in neural information processing systems*, 32, 2019.

Reuven Y Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997.

Junwon Seo, Kensuke Nakamura, and Andrea Bajcsy. Uncertainty-aware latent safety filters for avoiding out-of-distribution failures. In *Conference on Robot Learning*, pp. 4442–4472. PMLR, 2025.

Dhruv Shah, Benjamin Eysenbach, Gregory Kahn, Nicholas Rhinehart, and Sergey Levine. Rapid exploration for open-world navigation with latent goal models. *arXiv preprint arXiv:2104.05859*, 2021.

Dhruv Shah, Ajay Sridhar, Arjun Bhorkar, Noriaki Hirose, and Sergey Levine. Gnm: A general navigation model to drive any robot. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7226–7233. IEEE, 2023a.

Dhruv Shah, Ajay Sridhar, Nitish Dashora, Kyle Stachowicz, Kevin Black, Noriaki Hirose, and Sergey Levine. Vint: A foundation model for visual navigation. In *Conference on Robot Learning*, pp. 711–733. PMLR, 2023b.

Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics: Results of the 11th international conference*, pp. 621–635. Springer, 2017.

Wangtian Shen, Ziyang Meng, Jinming Ma, Mingliang Zhou, and Diyun Xiang. An efficient and multi-modal navigation system with one-step world model. *arXiv preprint arXiv:2601.12277*, 2026.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

Jaskirat Singh, Boyang Zheng, Zongze Wu, Richard Zhang, Eli Shechtman, and Saining Xie. Improved baselines with representation autoencoders. *arXiv preprint arXiv:2605.18324*, 2026.

Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. Nomad: Goal masked diffusion policies for navigation and exploration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 63–70. IEEE, 2024.

Michael E. Tipping and Christopher M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 61(3):611–622, 09 1999. ISSN 1369-7412. doi: 10.1111/1467-9868.00196.

Hanqing Wang, Wei Liang, Luc Van Gool, and Wenguan Wang. Dreamwalker: Mental planning for continuous vision-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10873–10883, 2023.

Yunheng Wang, Yuetong Fang, Taowen Wang, Yixiao Feng, Yawen Tan, Shuning Zhang, Peiran Liu, Yiding Ji, and Renjing Xu. Dreamnav: A trajectory-based imaginative framework for zero-shot vision-and-language navigation. *arXiv preprint arXiv:2509.11197*, 2025.

Qiaoyun Wu, Jun Wang, Jing Liang, Xiaoxi Gong, and Dinesh Manocha. Image-goal navigation in complex environments via modular learning. *IEEE Robotics and Automation Letters*, 7(3): 6902–6909, 2022.

Wei Xu and Fu Zhang. Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter. *IEEE Robotics and Automation Letters*, 6(2):3317–3324, 2021.

Han Yan, Zishang Xiang, Zeyu Zhang, and Hao Tang. Mwm: Mobile world models for action-conditioned consistent prediction. *arXiv preprint arXiv:2603.07799*, 2026.

Zichen Yan, Rui Huang, Lei He, Shao Guo, and Lin Zhao. Sign: Safety-aware image-goal navigation for autonomous drones via reinforcement learning. *IEEE Robotics and Automation Letters*, 11 (2):1962–1969, 2025.

Hang Yin, Xiuwei Xu, Linqing Zhao, Ziwei Wang, Jie Zhou, and Jiwen Lu. Unigoal: Towards universal zero-shot goal-oriented navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19057–19066, 2025.

Yiming Zeng, Hao Ren, Shuhang Wang, Junlong Huang, and Hui Cheng. Navidiffusor: Cost-guided diffusion model for visual navigation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11994–12001. IEEE, 2025.

Mingkun Zhang, Wangtian Shen, Fan Zhang, Haijian Qin, Zihao Pei, and Ziyang Meng. Rae-nwm: Navigation world model in dense visual representation space. *arXiv preprint arXiv:2603.09241*, 2026.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

Tianyao Zhang, Xiaoguang Hu, Jin Xiao, and Guofeng Zhang. A survey of visual navigation: From geometry to embodied ai. *Engineering Applications of Artificial Intelligence*, 114:105036, 2022.

Weichen Zhang, Peizhi Tang, Xin Zeng, Fanhang Man, Shiquan Yu, Zichao Dai, Baining Zhao, Hongjin Chen, Yu Shang, Wei Wu, et al. Aerial world model for long-horizon visual generation and navigation in 3d space. *arXiv preprint arXiv:2512.21887*, 2025a.

Yuhang Zhang, Haosheng Yu, Jiaping Xiao, and Mir Feroskhan. Grounded vision-language navigation for uavs with open-vocabulary goal understanding. *arXiv preprint arXiv:2506.10756*, 2025b.

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 e-prints*, pp. arXiv–2511, 2025c.

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*, 2025d.

Baining Zhao, Jiacheng Xu, Weicheng Feng, Xin Zhang, Zhaolu Wang, Haoyang Wang, Shilong Ji, Ziyou Wang, Jianjie Fang, Zhiheng Zheng, et al. Worldvln: Autoregressive world action model for aerial vision-language navigation. *arXiv preprint arXiv:2605.15964*, 2026.

Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. *arXiv preprint arXiv:2510.11690*, 2025.

Gaoyue Zhou, Hengkai Pan, Yann Lecun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. In *International Conference on Machine Learning*, pp. 79115–79135. PMLR, 2025.

Deyi Zhu, Yuji Wang, Yong Liu, Yansong Tang, Bingyao Yu, Jiwen Lu, and Jie Zhou. Segment anything with motion, geometry, and semantic adaptation for complex nonlinear visual object tracking. *arXiv preprint arXiv:2605.22538*, 2026.

Yixuan Zhu, Jiaqi Feng, Wenzhao Zheng, Yuan Gao, Xin Tao, Pengfei Wan, Jie Zhou, and Jiwen Lu. Astra: General interactive world model with autoregressive denoising. *arXiv preprint arXiv:2512.08931*, 2025.

Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J. Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3357–3364. IEEE, 2017.

A Appendix

A.1 Additional Related Work

A.1.1 Uncertainty-Aware Visual Navigation

Uncertainty in visual navigation has been studied primarily in current perception and semantic representation. Existing methods calibrate and temporally aggregate segmentation predictions for object navigation (Prasanna et al., 2024), use vision–language uncertainty to guide active object search (Bajpai et al., 2025), or integrate geometric, semantic, and appearance uncertainty into 3D Gaussian maps for vision-language navigation (Gao et al., 2026a). Others estimate model familiarity or action confidence to detect out-of-distribution observations, invoke additional reasoning, or request assistance (Pohland & Tomlin, 2025; Ding et al., 2025; Igbinedion & Karaman, 2024).

A complementary line of work considers uncertainty in localization, mapping, and motion planning. Perception-aware planners select trajectories that reduce visual localization uncertainty (Costante et al., 2016), while UrbanFly (Harithas et al., 2023) models uncertainty in monocular visual–inertial maps for collision-aware UAV planning. Other approaches propagate state-estimation and model uncertainty into collision prediction and action selection (Nguyen et al., 2024), or construct calibrated uncertainty sets from noisy visual predictions for robust task-level optimization (Pan et al., 2026).

These methods primarily address uncertainty in current perception, localization, semantic mapping, or planning constraints. In contrast, we focus on *future-state uncertainty*: given the same visual context and candidate trajectory, multiple future observations may be plausible.

A.1.2 OOD Detection

Out-of-distribution (OOD) detection aims to identify test samples that deviate from the distribution represented by a model. Conventional approaches derive OOD scores from classifier confidence, energy functions, or distances in a learned feature space (Hendrycks & Gimpel, 2016; Lee et al., 2018; Liu et al., 2020). Generative approaches instead use likelihood, reconstruction error, or predictive uncertainty to determine whether an observation is supported by the learned data distribution (Nalisnick et al., 2018; Ren et al., 2019). However, these methods are primarily designed to detect distribution shifts in static inputs and do not account for how future observations depend on an agent's context and actions.

Recent studies have incorporated uncertainty and OOD detection into world models to identify unreliable predictions or unseen hazards (Seo et al., 2025; Mei et al., 2025). Their primary objective is to determine whether a world model is operating outside its training distribution, rather than to evaluate whether a visual goal is compatible with the future induced by a particular candidate trajectory.

A.2 Implementation Details of UA-NWM

This section provides further details on the architecture, training and inference procedures, and hyperparameter settings of UA-NWM.

A.2.1 Model Structure

UA-NWM consists of a lightweight deterministic latent world model and the proposed Hierarchical Error Projection (HEP) module. Their overall architectures are illustrated in Figure 3(c) and Figure 4(a), respectively. We provide the detailed structure of each component below.

DINO encoder Each RGB observation is resized to 224×224 and encoded by a frozen DINOv3 ViT-B/16 encoder (Siméoni et al., 2025). We discard the class and register tokens and retain the 14×14 patch grid, yielding $N = 196$ patch tokens with feature dimension $D = 768$:

$$x_t = \phi(o_t) \in \mathbb{R}^{N \times D}. \tag{16}$$

Token compressor. The compressor maps the dense DINO feature representation into $K = 32$ compact latent tokens, each with dimension $d = 384$:

$$z_t = C_\eta(x_t) \in \mathbb{R}^{K \times d}. \tag{17}$$

It comprises K learnable queries of dimension 384 and a six-head cross-attention layer. The normalized 768-dimensional DINO patch tokens are used as keys and values and are projected to the 384-dimensional attention space inside the cross-attention layer. Let $Q_C \in \mathbb{R}^{K \times 384}$ denote the learned compressor queries. The compression process is formulated as

$$h_t^C = \mathrm{MHA}_C\big(\mathrm{LN}(Q_C), \mathrm{LN}(x_t), \mathrm{LN}(x_t)\big), \tag{18}$$

$$z_t = h_t^C + \mathrm{MLP}_C\big(\mathrm{LN}(h_t^C)\big), \tag{19}$$

where the three inputs to MHA_C correspond to the queries, keys, and values, respectively. The feed-forward network MLP_C consists of $\mathrm{Linear}(384, 384)$, GELU, and $\mathrm{Linear}(384, 384)$. All dropout rates in the token compressor are set to zero.

Delta encoder and decoder. Following the delta-aware transition parameterization of DeltaWorld (Kerssies et al., 2026), we use $M = 32$ delta tokens, each with dimension 384, to represent the transition between two adjacent frames. During representation training, a teacher delta encoder extracts the latent transition from two adjacent compressed states. For each token slot, it concatenates the current state, next state, and their discrepancy,

$$u_{t+1} = [z_t; z_{t+1}; z_{t+1} - z_t] \in \mathbb{R}^{K \times 3d}, \tag{20}$$

and projects u_{t+1} from 1152 to 384 dimensions. The projected tokens are processed by two pre-norm Transformer encoder layers, each containing 6-head self-attention and an MLP with dimensions $384 \to 1536 \to 384$ and GELU activation. A final LayerNorm and linear projection produce

$$\delta_{t+1} = E_\Delta(z_t, z_{t+1}) \in \mathbb{R}^{32 \times 384}. \tag{21}$$

The delta encoder is used only to provide teacher delta tokens during representation training and is not required at inference.

The delta decoder has the same two-layer Transformer structure. It first concatenates z_t and a delta latent token-wise and projects the resulting 768-dimensional features to 384 dimensions. After two pre-norm Transformer layers and a final LayerNorm-linear projection, it predicts a residual update that is added to the current state:

$$D_\Delta(z_t, \delta) = z_t + W_\Delta \,\mathrm{LN}\big(\mathrm{Tr}_\Delta([z_t; \delta])\big). \tag{22}$$

Here, $\delta \in \mathbb{R}^{M \times 384}$ denotes the input delta tokens, $[\cdot;\cdot]$ denotes token-wise feature concatenation, Tr_Δ denotes the Transformer decoder, and W_Δ is the final linear projection.

Action-conditioned causal transformer. Given the compressed context states and a future action sequence, UA-NWM autoregressively applies a one-step transition at each action step. The normalized four-dimensional action a_t is embedded as

$$\psi(a_t) = W_2 \,\mathrm{SiLU}(W_1 a_t) \in \mathbb{R}^{384}, \tag{23}$$

where W_1 and W_2 denote linear projections, and both the hidden and output dimensions are 384. The predictor receives $C = 4$ compressed context states. Learned frame and token positional embeddings are added before the $4 \times 32 = 128$ context tokens are flattened and processed by six causal Transformer blocks. Each block contains 6-head self-attention and an MLP with dimensions $384 \to 1536 \to 384$ and GELU activation. The attention mask is causal across frames while allowing all tokens within the same frame to interact.

The action embedding modulates every Transformer block through adaptive LayerNorm (AdaLN). Each block uses an independent SiLU-linear modulation layer that maps the 384-dimensional action embedding to six 384-dimensional vectors: shift, scale, and residual gate parameters for the attention and MLP branches. The modulation output layer is zero-initialized. After the six blocks, only the 32 tokens associated with the most valuable information are retained and passed through a final LayerNorm and linear head to predict the delta latent:

$$\hat{\delta}_{t+1} = P_\omega\big(Z_{t-C+1:t}, \psi(a_t)\big) \in \mathbb{R}^{32 \times 384}. \tag{24}$$

The next absolute latent state is then composed by the delta decoder,

$$\hat{z}_{t+1} = D_\Delta(z_t, \hat{\delta}_{t+1}). \tag{25}$$

The predicted state is appended to the context and the oldest state is removed before processing the next action, yielding an autoregressive rollout over $A_{t:t+H-1}$.

Token fuser. The token fuser decodes a predicted compressed state into a dense DINO-space feature map. It starts from 196 learned 384-dimensional output queries and applies two cross-attention blocks over the 32 compressed tokens. Each block contains pre-normalized 6-head cross-attention, a residual connection, and a pre-normalized MLP with dimensions $384 \to 1536 \to 384$ and GELU activation. A final LayerNorm produces the shared per-patch hidden field $h_{t+1} \in \mathbb{R}^{196\times 384}$, and a linear readout maps it to the deterministic mean prediction

$$\mu_{t+1} = G_\zeta(\hat{z}_{t+1}) \in \mathbb{R}^{196\times 768}. \tag{26}$$

The deterministic baseline scores a candidate trajectory using the flattened cosine distance between its final-step mean prediction and the DINO feature map of the goal image. HEP uses the same output from the fuser , ensuring that the deterministic and uncertainty-aware variants share an identical predictive backbone.

Hierarchical Error Projection module. HEP predicts context- and trajectory-conditioned directions along which the true future feature may plausibly deviate from the deterministic prediction. Its basis predictor takes the fuser output and the compressed observation context $Z_{\text{ctx}} \in \mathbb{R}^{(CK)\times d}$ as input, where $C = 4$, $K = 32$, and thus $CK = 128$.

The goal feature is not involved in basis prediction; it is introduced only when the goal–prediction discrepancy is projected onto the subspace spanned by the predicted bases. Therefore, the uncertainty subspace is determined solely by the observation context and candidate trajectory through the rolled-out latent state, independently of the goal image.

HEP operates over a spatial pyramid. At scale g, the 14×14 patch grid is partitioned into g^2 cells and average-pooled within each cell, producing a cell descriptor

$$m_{g,c} = \frac{1}{|\Omega_{g,c}|} \sum_{i\in\Omega_{g,c}} h_i \in \mathbb{R}^{384}. \tag{27}$$

The four scales contain 1, 4, 49, and 196 cells, corresponding to 196, 49, 4, and 1 DINO patches per cell, respectively.

Each scale has an independent 4-head context cross-attention layer. The cell descriptors are used as queries, while the full 128 compressed context tokens are used as keys and values:

$$q_{g,c} = \text{MHA}_g\big(\text{LN}_g(m_{g,c}), \text{LN}(Z_{\text{ctx}}), \text{LN}(Z_{\text{ctx}})\big) \in \mathbb{R}^{384}. \tag{28}$$

The cell descriptor and attended context descriptor are concatenated and passed through a scale-specific basis MLP,

$$U_{g,c} = f_g([m_{g,c}, q_{g,c}]) \in \mathbb{R}^{D\times R}, \tag{29}$$

where $R = 2$, $D = 768$, and each f_g has the structure $768 \to 384 \to 384 \to 1536$ with GELU after the first two linear layers. The final 1536-dimensional output represents two 768-dimensional basis directions. The four scales use separate attention and MLP parameters.

Given the normalized goal discrepancy e, HEP performs a one-pass coarse-to-fine residual decomposition. Let r denote the residual entering a scale, initialized as $r = e$. For each cell, HEP computes the cell-average residual $\bar{r}_{g,c}$ and obtains ridge-regularized projection coefficients

$$\alpha_{g,c} = \big(U_{g,c}^\top U_{g,c} + \lambda I\big)^{-1} U_{g,c}^\top \bar{r}_{g,c}, \tag{30}$$

where $\lambda = 10^{-3}$. The explained component $p_{g,c} = U_{g,c}\alpha_{g,c}$ is subtracted from every patch residual in the cell. The resulting residual is then passed to the next finer scale. Coarse scales remove globally or regionally coherent deviations, whereas the finest 14×14 scale performs patch-wise low-rank projection. The basis vectors are not explicitly orthogonalized; the Gram-matrix solve above projects onto their span and remains stable when the predicted directions are correlated.

After the finest scale, the remaining field is the hierarchically unexplained residual $e^\perp$. This procedure is a sequential ridge projection over the scale pyramid, rather than a single global orthogonal projection onto the union of all scale-wise subspaces.

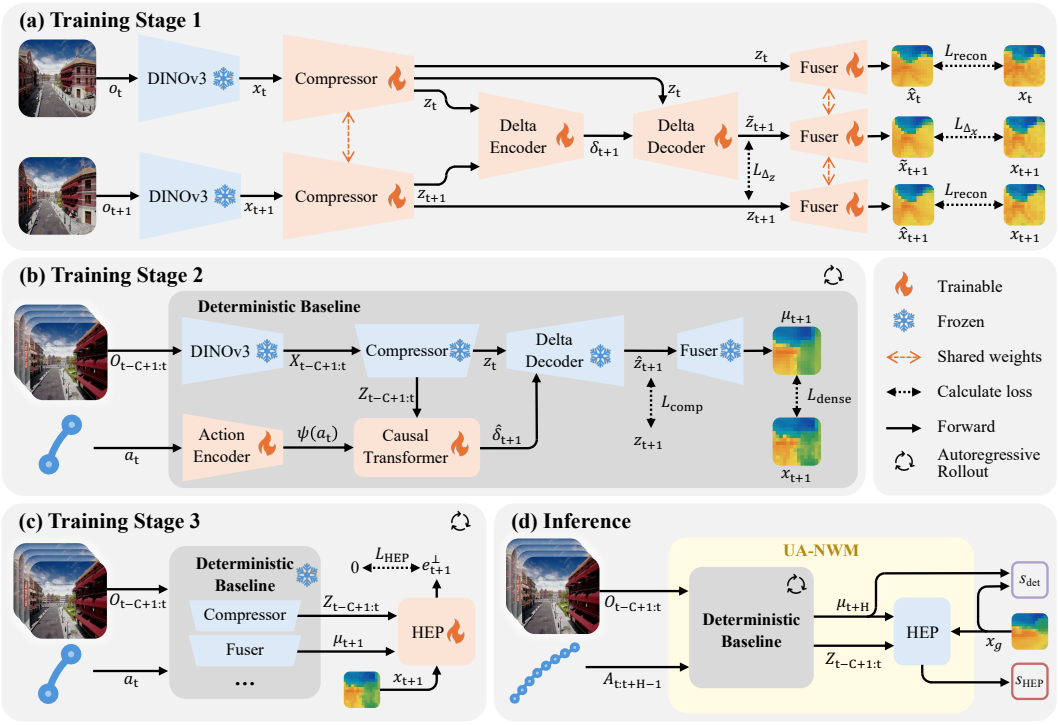


Figure 7: Detailed training and inference procedure of UA-NWM.

A.2.2 Training and Inference Procedure

UA-NWM is trained in three stages, as illustrated in Figure 7. The first two stages learn the deterministic latent world model baseline, and the final stage learns the uncertainty-aware HEP module.

Stage 1: representation training. We first train the representation modules, including the compressor, fuser, delta encoder, and delta decoder. Given an input feature map x_t, the compressor produces $z_t = C_\eta(x_t)$, and the fuser reconstructs $\hat{x}_t = G_\zeta(z_t)$. The basic reconstruction objective is

$$\mathcal{L}_{\text{recon}} = \|\hat{x}_t - x_t\|_2^2 + \beta_{\text{rec}} \left(1 - \cos(\hat{x}_t, x_t)\right). \tag{31}$$

This stage also trains the delta encoder and decoder. Given two adjacent compressed states z_t and z_{t+1}, the delta encoder extracts transition latent δ_{t+1}, and the delta decoder reconstructs the next compressed state:

$$\delta_{t+1} = E_\Delta(z_t, z_{t+1}), \quad \tilde{z}_{t+1} = D_\Delta(z_t, \delta_{t+1}). \tag{32}$$

We decode the reconstructed state as $\tilde{x}_{t+1} = G_\zeta(\tilde{z}_{t+1})$. This transition is supervised in both compressed latent space and dense DINO space:

$$\mathcal{L}_{\Delta z} = \|\tilde{z}_{t+1} - z_{t+1}\|_2^2, \tag{33}$$

$$\mathcal{L}_{\Delta x} = \|\tilde{x}_{t+1} - x_{t+1}\|_2^2 + \rho \left(1 - \cos(\tilde{x}_{t+1}, x_{t+1})\right). \tag{34}$$

The objective of stage 1 is:

$$\mathcal{L}_{\text{rep}} = \gamma_1 \cdot \mathcal{L}_{\text{recon}} + \gamma_2 \cdot \mathcal{L}_{\Delta z} + \gamma_3 \cdot \mathcal{L}_{\Delta x}, \tag{35}$$

where γ_1, γ_2 and γ_3 are scalar weights applied to each term in implementation.

Stage 2: deterministic prediction training. We then train the action encoder and action-conditioned causal Transformer with the compressor, fuser and delta decoder fixed. The prediction is supervised by the DINO feature and compressed latent of the target:

$$\mathcal{L}_{\text{dense}} = \|\mu_{t+1} - x_{t+1}\|_2^2 + \beta_{\text{pred}} \left(1 - \cos(\mu_{t+1}, x_{t+1})\right), \tag{36}$$

$$\mathcal{L}_{\text{comp}} = \lambda_{\text{comp}} \|\hat{z}_{t+1} - z_{t+1}\|_2^2, \tag{37}$$

$$\mathcal{L}_{\text{pred}} = \mathcal{L}_{\text{dense}} + \mathcal{L}_{\text{comp}}. \tag{38}$$

Stage 3: HEP training. Finally, we freeze the deterministic baseline and train only HEP. Given the predicted dense feature μ_{t+1} and target feature x_{t+1}, HEP predicts the uncertainty subspace and computes the remaining residual $e^{\perp}_{t+1}$. The objective minimizes the normalized residual fraction:

$$\mathcal{L}_{\text{HEP}} = \frac{\sum_{i=1}^{N} \left\| e^{\perp}_{t+1,i} \right\|_2^2}{\sum_{i=1}^{N} \left\| e_{t+1,i} \right\|_2^2 + \epsilon}. \tag{39}$$

This loss is magnitude-invariant and encourages HEP to learn directions along which the true future may plausibly deviate from the deterministic mean.

Although the objectives above are written for a single transition for clarity, UA-NWM uses an autoregressive eight-step rollout for both stage 2 and stage 3 training in order to mitigate error accumulation. The same losses are applied to every predicted step and averaged over time.

During training, we enable short-context augmentation. When a sampled window is close to the beginning of a trajectory with fewer than $C = 4$ real context frames available, the missing front context slots are filled by repeating the earliest available frame. The corresponding repeated-frame action intervals become zero-motion actions. This exposes the model to deficient context windows that may occur at the beginning of closed-loop simulation or deployment.

Inference. At inference time, UA-NWM consists of the deterministic latent baseline as backbone and the HEP module as a scoring head. Given a candidate action sequence $A_{t:t+H-1}$, the backbone first encodes the observation context into $Z_{t-C+1:t}$ and autoregressively predicts future compressed states. At each rollout step, the predictor estimates a delta latent from the current latent context and the corresponding action window; the delta decoder composes the next compressed state, which is then appended to the context for the following step. After H steps, the rollout returns predicted states $\hat{z}_{t+1:t+H}$. Candidate selection uses only the final state, whose fuser output is the final-horizon dense DINO prediction $\mu_{t+H} = G_\zeta(\hat{z}_{t+H})$, because the goal image specifies the desired terminal observation.

The deterministic baseline and UA-NWM share the same backbone and differ only in the scoring mechanism. The deterministic baseline directly compares the mean prediction with the goal DINO feature using flattened cosine distance:

$$s_{\text{det}} = 1 - \cos(\text{vec}(\mu_{t+H}), \text{vec}(x_g)) \,. \tag{40}$$

UA-NWM instead applies HEP as a final-step scoring head after the autoregressive rollout has finished. HEP predicts the uncertainty subspace from the deterministic prediction and the compressed context tokens; the goal feature is used only when projecting the final discrepancy, and the HEP output is not fed back into the rollout dynamics. Let

$$e_{t+H} = \text{norm}(x_g) - \text{norm}(\mu_{t+H}), \tag{41}$$

and let $e^{\perp}_{t+H}$ be the residual left by the HEP decomposition. The trajectory cost is the unnormalized residual energy:

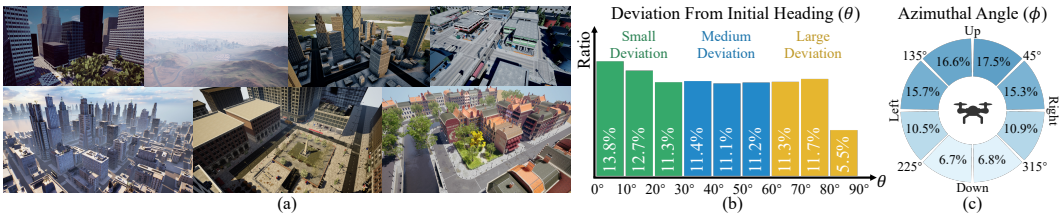
$$s_{\text{HEP}} = \frac{1}{N} \sum_{i=1}^{N} \left\| e^{\perp}_{t+H,i} \right\|_2^2 . \tag{42}$$

This differs from the normalized training loss: at test time, retaining the absolute magnitude gives sharper discrimination between candidate trajectories. In offline ranking, the candidate with the lowest score is selected from the fixed candidate set. In CEM planning, the same score is used as the optimization cost for sampled action sequences.

A.2.3 Hyper-parameter Settings

Unless otherwise stated, all training stages use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, bf16 automatic mixed precision, gradient clipping with norm 1.0, and a linear warmup followed by cosine decay. The action representation is a 4-dimensional local-frame delta pose. Position deltas are normalized by the waypoint spacing of 5.0 meters, and the action normalizer is reused for both training and evaluation.

For training stage 1, we use a per-GPU batch size of 12 under 4-GPU DDP, giving an effective batch size of 48. Stage 1 is trained for 10k steps with learning rate 10^{-4}, weight decay 0.05, 600



(a) (b) (c)

Figure 8: Overview of the AirGoal-10k benchmark. (a) Diverse navigation environments. (b) Distribution of deviations from the initial heading. (c) Distribution of azimuthal angles.

warmup steps, and minimum learning-rate ratio 0.01. The loss weights are $\beta_{\text{rec}} = 0.05$, $\rho = 0.2$, and $\gamma_1 = \gamma_2 = \gamma_3 = 1.0$. We inject Gaussian noise with standard deviation 0.01 into compressed latents and 0.005 into teacher delta tokens during this stage.

For training stage 2, the rollout horizon is 8. The predictor is trained for 24k steps with learning rate 10^{-4}, weight decay 0.05, 1000 warmup steps, and minimum learning-rate ratio 0.01. The dense DINO loss uses $\beta_{\text{pred}} = 0.05$, and the compressed-latent supervision weight is $\lambda_{\text{comp}} = 0.1$.

For training stage 3, the deterministic baseline is initialized from the best prediction checkpoint and kept frozen. HEP is trained on a single GPU with batch size 48 for 8k steps, using learning rate 5×10^{-4}, 300 warmup steps, minimum learning-rate ratio 0.02, and no weight decay. The rollout horizon is also 8, and the HEP loss is averaged over all predicted steps. For the normalized HEP cost, the denominator clamp uses $\epsilon = 10^{-9}$.

A.3 Details of AirGoal-10k Benchmark

A.3.1 Dataset Construction

We construct a UAV image-goal navigation benchmark from AirSim (Shah et al., 2017), with the goal of learning first-person visual dynamics conditioned on future motion.

Start points. Instead of uniformly sampling arbitrary poses in AirSim, which often produces invalid viewpoints inside geometry, below the ground, or at unrealistic altitudes, we sample start states from trajectory annotations of existing aerial VLN benchmarks (Liu et al., 2023; Gao et al., 2025). This reuses poses that are already near feasible navigation regions while still allowing us to resample new future motions. Start states are deduplicated by scene, position, and yaw, and the held-out validation/test splits are constructed so that their start states do not overlap with the training split.

Future trajectory sampling. In our preliminary analysis, directly cropping short segments from the original VLN trajectories led to a highly imbalanced turning distribution: most segments had a heading-change angle below $15°$. Such data would make the learned world model accurate for nearly straight motion but unreliable under large turns, which are important for closed-loop navigation. We therefore use the original trajectories only as feasible start-state pools, and resample the future motion ourselves to cover a broad range of turn magnitudes and directions.

For each selected start state, we generate future trajectories by sampling a target direction in the local coordinate frame of the UAV. Let the start position and yaw be $p_0 = (x_0, y_0, z_0)$ and ψ_0. We define the local forward, right, and upward directions as

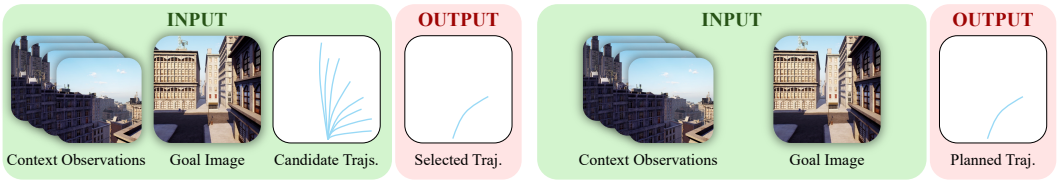
$$\begin{aligned} e_f &= (\cos\psi_0, \sin\psi_0, 0), \\ e_r &= (-\sin\psi_0, \cos\psi_0, 0), \\ e_u &= (0, 0, -1), \end{aligned} \tag{43}$$

where e_u follows the AirSim NED convention, in which negative z corresponds to upward motion. For each trajectory, we sample

$$\theta \sim \mathcal{U}(0, 85°), \quad \varphi \sim \mathcal{U}(0, 2\pi), \tag{44}$$

and construct the target direction

$$d_{\text{target}} = \cos\theta\, e_f + \sin\theta\cos\varphi\, e_r + \sin\theta\sin\varphi\, e_u. \tag{45}$$



(a) Trajectory Ranking (b) Standalone Planning

Figure 9: Comparison of (a) trajectory ranking and (b) standalone planning tasks of the AirGoal-10k benchmark.

The angle θ controls the deviation from the initial heading, while the azimuthal angle φ determines whether this deviation is left/right, upward/downward, or a combined 3D turn.

Given d_{target}, the segment direction at each timestep is obtained by spherical interpolation from e_f to d_{target} with a smoothstep schedule. The UAV position is then generated by forward integration:

$$\begin{aligned} p_{i+1} &= p_i + s_i d_i, \\ s_i &= s_0(1+\epsilon_i), \\ \epsilon_i &\sim \mathcal{U}(-0.1, 0.1). \end{aligned} \tag{46}$$

The base step size s_0 is set to match the average motion interval observed in the existing VLN trajectories. In our implementation this corresponds to approximately five meters per frame interval. Roll and pitch are fixed to zero, and yaw is updated from the horizontal projection of the instantaneous motion direction.

Rendering and filtering. AirSim is launched scene-by-scene and each accepted trajectory is rendered from the front RGB camera. AerialVLN (Liu et al., 2023) scenes are rendered directly at 512×512. OpenFly (Gao et al., 2025) scenes are first rendered at high resolution and then center-cropped and resized to 512×512. We apply automatic quality filters during collection, including invalid-height checks, black-frame rejection, and an LPIPS-based (Zhang et al., 2018) threshold for crash detection.

Dataset composition and distribution. The final dataset contains 11,000 trajectories: 9,000 training trajectories, 1,000 validation trajectories, and 1,000 test trajectories. As shown in Figure 8, these trajectories span diverse urban navigation environments. The distribution of the initial-heading deviation θ is approximately uniform over $[0, 90^\circ]$, while the azimuth angle φ covers the full range of $[0, 360^\circ]$. Together, these distributions preserve diverse left–right and up–down motion components for trajectory-conditioned future prediction.

A.3.2 Task Definition

AirGoal-10k supports two offline evaluation tasks, as illustrated in Figure 9. Both tasks start from the same image-goal navigation input: the agent observes a context of $C = 4$ RGB frames and is given a goal image.

Trajectory ranking. In trajectory ranking, a set of candidate trajectories are provided before scoring. In our test split, each case contains 32 externally generated candidate trajectories stored with the trajectory metadata. These candidates share the same observed context but branch into different 8-step future waypoint sequences. Each method scores the same candidates for fairness and selects

$$A^\star = \underset{A \in \mathcal{A}_N}{\operatorname{argmin}}\, s(A, o_g \mid O_{t-C+1:t}), \tag{47}$$

where $\mathcal{A}_N$ denotes the first N candidates and $N \in \{8, 16, 32\}$. This task evaluates the quality of the trajectory scoring function under a fixed proposal set.

Standalone planning. Standalone planning removes the external candidate set. Policy based methods can directly predict one or several trajectories, while world-model based methods use CEM following NWM (Bar et al., 2025). At each iteration, the planner samples a set of action sequences,

rolls out and scores them with the world model, keeps the lowest-cost elites, and refits the sampling distribution over trajectory parameters. After the last iteration, the planned trajectory is generated from the mean parameter of the final CEM distribution. This setting tests whether the scorer can guide search without relying on a proposal policy. In our implementation, CEM uses 32 samples, the top 16 elites, and 3 optimization iterations.

For both tasks, the selected or planned trajectory is compared with the ground-truth future path using ATE and RPE.

A.4 Experimental Details

A.4.1 Baseline Implementation

We compare our method with representative policy-based and world-model-based baselines. The policy-based methods include GNM, ViNT, NoMaD, FlowNav, and NaviBridger (Shah et al., 2023a;b; Sridhar et al., 2024; Gode et al., 2025; Ren et al., 2025), which directly predict actions or trajectories from visual observations. The world-model-based methods include NWM, One-Step WM, MWM, and RAE-NWM (Bar et al., 2025; Shen et al., 2026; Yan et al., 2026; Zhang et al., 2026), which predict future observations or latent states to rank candidate trajectories.

Since these previous baselines are originally designed for 2D navigation, we modify their action encoding modules to adapt them to 3D space. For methods with different model size options, we select variants with similar parameter counts for fair comparison. For NWM (Bar et al., 2025) and MWM (Yan et al., 2026), we adopt CDiT-B/2. For RAE-NWM (Zhang et al., 2026), we adopt CDiT-B/2 and DINOv2-B (Oquab et al., 2024).

All baselines are trained on the training set of AirGoal-10k until full convergence. For NaviBridger (Ren et al., 2025), we additionally train its CVAE model on AirGoal-10k in advance and use it as the learning-based prior. For One-Step WM (Shen et al., 2026), the official implementation includes a pretraining strategy, but the corresponding pretrained checkpoint has not been released. Therefore, we train it from scratch.

During inference, policy-based methods use diffusion or flow matching to generate trajectories, and we follow their default sampling steps. NoMaD (Sridhar et al., 2024) and NaviBridger (Ren et al., 2025) use 10-step diffusion, while FlowNav (Gode et al., 2025) uses 10-step flow matching ODE. In contrast, world models use diffusion or flow matching to generate future observations. For NWM (Bar et al., 2025), the official setting with 250 sampling steps is prohibitively slow for practical applications (approximately 30 minutes per step in online simulation experiments). Therefore, following MWM (Yan et al., 2026), we use 25 diffusion steps for NWM, reducing its latency to a comparable level with others. For other world models, we follow their official settings: MWM (Yan et al., 2026) uses 5 diffusion steps, RAE-NWM (Zhang et al., 2026) uses 50-step flow matching ODE, and One-Step WM (Shen et al., 2026) uses one-step shortcut flow matching.

A.4.2 Offline Experiments

All offline experiments are conducted on 4 NVIDIA RTX 4090 GPUs with 24GB memory each. All speed tests are measured on a single GPU with the same batch size.

For the trajectory ranking task, we additionally report random selection and oracle selection strategies as references. Oracle selection always selects the trajectory with the smallest ATE with respect to the ground-truth trajectory.

A.4.3 Online Simulation Experiments

All online experiments are conducted on 4 NVIDIA RTX 3090 GPUs with 24GB memory each.

We evaluate online navigation on 100 start–goal pairs collected from AirSim (Shah et al., 2017), including 57 AerialVLN (Liu et al., 2023) episodes and 43 OpenFly (Gao et al., 2025) episodes. The start-to-goal distance is computed as the three-dimensional Euclidean distance between the initial UAV position and the goal location. The average distance is 57 m. Among the 100 episodes, 76%

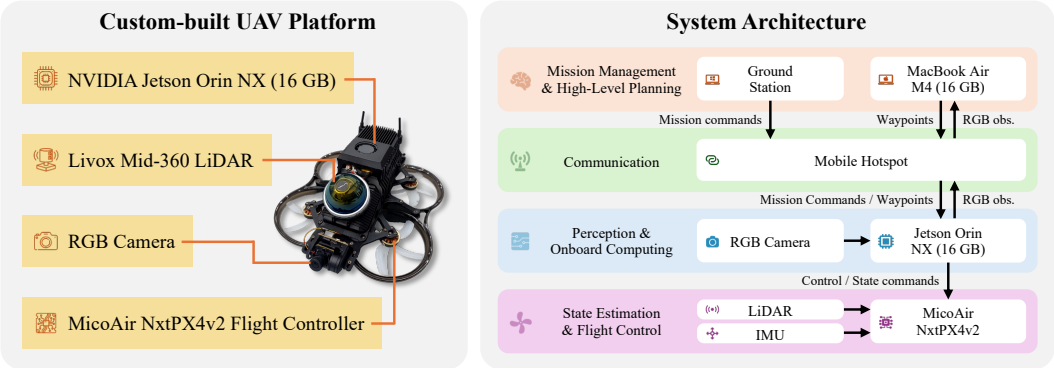


Figure 10: Configuration of custom-built UAV platform and system architecture for real-world experiments.

have distances between 45 m and 60 m, while the remaining 24% are longer-range cases between 60 m and 80 m.

Navigation is performed in a closed-loop manner. At each step, the UAV captures a new front-camera observation, plans its trajectory conditioned on the context observations and the goal image, and executes only the first action of the predicted trajectory. The maximum navigation budget is 20 executed actions.

An episode is considered successful if the UAV's final three-dimensional distance to the ground-truth goal is no greater than 20 m at termination. Observation-capture failures and collisions are recorded as separate failure modes.

We report Success Rate (SR) and Success weighted by Path Length (SPL). SPL is computed using the initial start-to-goal distance as the reference distance and the accumulated flown distance as the executed path length:

$$\mathrm{SPL} = \frac{1}{N}\sum_{i=1}^{N} S_i \frac{L_i}{\max(L_i, P_i)}, \tag{48}$$

where N is the number of episodes, $S_i \in \{0, 1\}$ indicates whether episode i is successful, L_i denotes the initial start-to-goal distance, and P_i denotes the accumulated flown distance. For runtime evaluation, we report the average planner time per online step, excluding simulator startup, environment reset, and scene-switching overhead.

A.4.4 Real-world Experiments

To evaluate the deployability of our method in real-world environments, we develop a custom-built quadrotor platform equipped with an NVIDIA Jetson Orin NX (16 GB), a forward-facing RGB camera, a Livox Mid-360 LiDAR, and a MicoAir NxtPX4v2 flight controller, as illustrated in Figure 10. The RGB camera provides egocentric visual observations for image-goal navigation, while the Jetson Orin NX serves as the onboard computing and communication unit. The LiDAR measurements and onboard IMU data are fused by FAST-LIO (Xu & Zhang, 2021) to estimate the real-time position and orientation of the UAV. The resulting state estimates are provided to the low-level flight-control system, while the NxtPX4v2 flight controller is responsible for executing control commands and maintaining stable UAV motion.

The system adopts a distributed computation architecture. The UAV communicates with a Windows laptop ground station and a MacBook Air M4 (16 GB) through a mobile-phone hotspot. The ground station is used for mission management and flight supervision, whereas world model inference and high-level navigation planning is performed locally on the MacBook. In our experiments, the MacBook runs on battery power without being plugged into an external power supply. During navigation, the current RGB observation is captured by the onboard camera and forwarded through the Jetson Orin NX to the high-level planner. Conditioned on the current observation and the goal image, UA-NWM plans with CEM, sampling 32 candidate trajectories at each of three optimization

iterations, and sends the resulting waypoint command back to the UAV. The Jetson Orin NX subsequently converts the received waypoint into control commands and forwards them to the NxtPX4v2 flight controller for execution. This communication and control process is repeated in a closed-loop manner after each executed action. Navigation is terminated when the LPIPS (Zhang et al., 2018) distance between the current observation and the goal image falls below 0.5.

Importantly, the proposed navigation model does not directly access the LiDAR measurements, IMU states, FAST-LIO pose estimates, ground-truth goal location, or ground-truth distance to the goal. FAST-LIO is used only for low-level state estimation and flight control to ensure stable and safe motion execution. High-level navigation decisions are made solely from the egocentric RGB observations and the given goal image. This separation allows us to evaluate whether the proposed method can generate executable navigation decisions on a physical UAV without relying on privileged geometric information. All real-world goal images are captured by a mobile phone, whose camera intrinsics and viewpoint characteristics differ from the onboard UAV camera. This setting further tests the robustness of UA-NWM to camera-parameter gaps between goal images and onboard observations. Additional real-world visualizations are provided in Figure 18.

A.4.5 DINO Latent Visualization

The trajectory scoring process of UA-NWM and the deterministic baseline is performed directly in the DINOv3 (Siméoni et al., 2025) latent space. For visualizing the predicted DINOv3 latents, we adopt RAEv2 (Singh et al., 2026) and fine-tune the officially released checkpoint on the AirGoal-10k training set to improve reconstruction quality. The decoded RGB images are only used for qualitative visualization and may not faithfully represent all information contained in the predicted DINOv3 latents. Since DINOv3 is not optimized for pixel-level reconstruction, the decoded images may exhibit blurry details or color shifts, while still providing meaningful semantic and structural cues for qualitative analysis.

The sampling procedure for other plausible states within the predicted subspace is discussed in Section A.8.

A.5 Additional Ablation Studies and Controlled Comparisons

A.5.1 Rollout Steps for Training

As described in Section A.2.2, we employ autoregressive rollout in stages 2 and 3 and average the losses across all rollout steps to mitigate error accumulation in autoregressive world models. We vary the number of rollout steps to investigate its effect on navigation performance. As shown in Table 6, performance generally improves with longer rollouts. Exposing the model to its own intermediate predictions during training better matches autoregressive inference and improves robustness to accumulated errors. We therefore use an eight-step rollout in both Stages 2 and 3.

A.5.2 Staged Training and Delta Representation

As described in Section A.2.2, our deterministic backbone adopts a staged training strategy that first learns compact representations and latent transitions before learning future prediction. Following DeltaWorld (Kerssies et al., 2026), we further employ a delta representation to encode the change in visual representations between adjacent frames, allowing the predictor to estimate this delta rather than the full representation of the next frame. We ablate both designs in Table 7.

Training the deterministic backbone end-to-end without staged training degrades performance, indicating that pretraining the compressor, fuser, and delta decoder to model compact representations and latent transitions provides a more stable and effective initialization for future prediction. Replacing delta prediction with direct prediction of the full next-frame representation causes a larger performance drop across different numbers of candidate trajectories. This suggests that predicting inter-frame changes reduces the redundancy shared by adjacent observations and enables the predictor to focus on temporally varying information, thereby facilitating more accurate dynamics modeling. Combining both strategies yields the best overall performance.

Table 6: Comparison of different rollout steps for training.

Rollout Steps	8 Candidates		16 Candidates		32 Candidates	
	ATE ↓	RPE ↓	ATE ↓	RPE ↓	ATE ↓	RPE ↓
1	1.373	0.378	1.296	0.358	1.292	0.357
2	1.319	0.364	1.244	0.345	1.185	0.329
3	1.298	0.358	1.218	0.337	1.155	0.322
4	1.301	0.358	1.197	0.332	1.134	0.315
5	1.297	0.357	1.203	0.333	1.132	0.314
6	1.286	0.354	1.198	0.332	1.114	0.310
7	1.281	0.354	1.199	0.332	1.123	0.312
8	**1.256**	**0.347**	**1.174**	**0.326**	**1.091**	**0.304**

Table 7: Ablation study on the staged training strategy and delta representation.

Staged Training	Delta Representation	8 Candidates		16 Candidates		32 Candidates	
		ATE ↓	RPE ↓	ATE ↓	RPE ↓	ATE ↓	RPE ↓
×	✓	1.309	0.360	1.222	0.338	1.189	0.330
✓	×	1.370	0.375	1.309	0.359	1.263	0.347
✓	✓	**1.256**	**0.347**	**1.174**	**0.326**	**1.091**	**0.304**

Table 8: Comparison of different combinations of K and M.

K	M	8 Candidates		16 Candidates		32 Candidates	
		ATE ↓	RPE ↓	ATE ↓	RPE ↓	ATE ↓	RPE ↓
32	16	1.342	0.370	1.264	0.350	1.204	0.334
32	64	1.359	0.374	1.289	0.355	1.233	0.341
16	32	1.309	0.360	1.235	0.341	1.159	0.321
64	32	1.309	0.360	1.230	0.340	1.155	0.320
32	32	**1.256**	**0.347**	**1.174**	**0.326**	**1.091**	**0.304**

Table 9: Alternative uncertainty formulations using the same frozen mean predictor.

Uncertainty Formulation	Ranking (8)		Ranking (16)		Ranking (32)		Planning	
	ATE ↓	RPE ↓	ATE ↓	RPE ↓	ATE ↓	RPE ↓	ATE ↓	RPE ↓
Diagonal Mahalanobis	1.358	0.372	1.289	0.354	1.210	0.334	1.271	0.348
Low-rank PPCA ($R = 32$)	1.360	0.373	1.297	0.357	1.233	0.340	1.287	0.354
Heteroscedastic regression	1.330	0.367	1.271	0.351	1.217	0.336	1.302	0.362
PPCA Monte Carlo (32 samples)	1.351	0.371	1.290	0.355	1.226	0.339	1.280	0.351
HEP (Ours)	**1.256**	**0.347**	**1.174**	**0.326**	**1.091**	**0.304**	**1.222**	**0.334**

Table 10: Paired ATE difference between HEP and each uncertainty baseline. $\Delta = \mathrm{HEP} - \mathrm{baseline}$, so negative values favor HEP. Confidence intervals use 10,000 paired case-bootstrap resamples.

Baseline	Ranking (8 Candidates)		Ranking (16 Candidates)		Ranking (32 Candidates)		Planning	
	Δ	95% CI	Δ	95% CI	Δ	95% CI	Δ	95% CI
Mahalanobis	$\mathbf{-0.102}$	$[-0.134, -0.071]$	$\mathbf{-0.115}$	$[-0.146, -0.084]$	$\mathbf{-0.119}$	$[-0.149, -0.089]$	$\mathbf{-0.049}$	$[-0.069, -0.028]$
Low-rank PPCA	$\mathbf{-0.104}$	$[-0.138, -0.073]$	$\mathbf{-0.123}$	$[-0.157, -0.089]$	$\mathbf{-0.142}$	$[-0.176, -0.108]$	$\mathbf{-0.065}$	$[-0.087, -0.044]$
Heteroscedastic	$\mathbf{-0.074}$	$[-0.098, -0.052]$	$\mathbf{-0.097}$	$[-0.123, -0.071]$	$\mathbf{-0.126}$	$[-0.153, -0.099]$	$\mathbf{-0.080}$	$[-0.109, -0.053]$
PPCA Monte Carlo	$\mathbf{-0.095}$	$[-0.130, -0.062]$	$\mathbf{-0.116}$	$[-0.150, -0.083]$	$\mathbf{-0.135}$	$[-0.168, -0.100]$	$\mathbf{-0.058}$	$[-0.079, -0.037]$

Table 11: Effect of penalty on uncertainty-explainable discrepancy.

α	Ranking (8)		Ranking (16)		Ranking (32)		Planning	
	ATE ↓	RPE ↓	ATE ↓	RPE ↓	ATE ↓	RPE ↓	ATE ↓	RPE ↓
0	**1.256**	**0.347**	**1.174**	**0.326**	**1.091**	**0.304**	**1.222**	**0.334**
0.001	**1.256**	**0.347**	**1.174**	**0.326**	1.092	**0.304**	**1.222**	**0.334**
0.01	1.257	**0.347**	1.175	**0.326**	1.093	0.305	**1.222**	**0.334**
0.1	1.265	0.349	1.183	0.328	1.112	0.309	1.227	0.336
0.25	1.293	0.356	1.211	0.335	1.146	0.318	1.234	0.338
0.5	1.329	0.365	1.261	0.348	1.184	0.327	1.242	0.341
1.0	1.376	0.377	1.313	0.361	1.244	0.342	1.260	0.346

A.5.3 Number of Compressed and Delta Tokens

We further investigate the effects of K and M by evaluating different combinations of the two hyperparameters. As shown in Table 8, the setting $K = 32$ and $M = 32$ achieves the best performance across all candidate-set sizes. Deviating from this configuration in either direction consistently leads to inferior results, indicating that both representations benefit from a moderate capacity. Smaller values may limit their expressive power, whereas larger values may introduce unnecessary redundancy and make optimization more difficult. We therefore adopt $K = 32$ and $M = 32$ as the default setting.

A.5.4 Alternative Uncertainty Formulations

We compare HEP with four uncertainty-aware scoring formulations while holding the mean predictor fixed: (i) diagonal Mahalanobis scoring, following Gaussian feature-space uncertainty models (Lee et al., 2018), estimates global per-channel residual variance; (ii) rank-32 PPCA (Tipping & Bishop, 1999) models global low-rank covariance plus isotropic discarded variance; (iii) heteroscedastic regression (Nix & Weigend, 1994) predicts input-dependent variance and optimizes Gaussian negative log-likelihood, following the learned aleatoric formulation of Kendall & Gal (2017); and (iv) PPCA Monte Carlo (Tipping & Bishop, 1999) scores the nearest of 32 fixed common-random-number residual samples together with the PPCA reconstruction residual. All alternatives use $e = \mathrm{norm}(x_g) - \mathrm{norm}(\mu)$. They are fit using one full-context, horizon-eight endpoint from each of 9,000 training trajectories of AirGoal-10k. No test images, candidates, ATE, or RPE are used for fitting. The backbone is not retrained, but these alternatives fit either training-set statistics or a lightweight variance head. Tables 9 and 10 report aggregate performance and paired uncertainty, respectively.

HEP is best on both metrics for every task in Table 9. Moreover, every paired ATE interval in Table 10 excludes zero. The consistent advantage over global Mahalanobis and PPCA supports conditioning uncertainty on context and action, while the sampling variant shows that finite covariance sampling alone does not recover the benefit of hierarchical projection.

A.5.5 Penalty on Uncertainty-Explainable Discrepancy

To study how strongly discrepancies along the learned uncertainty directions should influence trajectory selection, we generalize the scoring function as

$$s_\alpha = \|e^\perp\|_2^2 + \alpha\|e^\parallel\|_2^2, \tag{49}$$

where $\alpha = 0$ is our score and $\alpha = 1$ equally penalizes both components. We evaluate UA-NWM with seven different weights.

As shown in Table 11, very small weights are effectively neutral at the reported precision, whereas larger weights progressively degrade both ranking and planning. No nonzero weight provides clear improvement, and equal treatment of the two components is consistently the weakest setting. This behavior supports the high-anisotropy assumption behind HEP: discrepancies captured by the learned directions should be strongly down-weighted for navigation scoring.

Table 12: Comparison of NWM, RAE-NWM and UA-NWM on the RECON and GO Stanford datasets using different numbers of trajectories generated by NoMaD.

Methods	RECON		GO Stanford	
	ATE ↓	RPE ↓	ATE ↓	RPE ↓
NWM (Bar et al., 2025) + NoMaD (×8)	1.01	0.27	2.06	0.59
NWM (Bar et al., 2025) + NoMaD (×16)	1.00	0.27	2.11	0.60
NWM (Bar et al., 2025) + NoMaD (×32)	0.99	0.27	2.11	0.60
RAE-NWM (Zhang et al., 2026) + NoMaD (×8)	1.07	0.30	1.97	0.57
RAE-NWM (Zhang et al., 2026) + NoMaD (×16)	1.09	0.31	2.02	0.58
RAE-NWM (Zhang et al., 2026) + NoMaD (×32)	1.11	0.32	2.01	0.59
UA-NWM (Ours) + NoMaD (×8)	0.94	**0.25**	1.68	0.49
UA-NWM (Ours) + NoMaD (×16)	**0.92**	**0.25**	1.52	**0.44**
UA-NWM (Ours) + NoMaD (×32)	**0.92**	**0.25**	**1.51**	**0.44**

A.6 Performance on 2D Navigation Benchmarks

We further evaluate UA-NWM on two 2D ground image-goal navigation benchmarks, RECON (Shah et al., 2021) and GO Stanford (Hirose et al., 2018), and compare it with NWM (Bar et al., 2025) and RAE-NWM (Zhang et al., 2026). We use the official inference settings of 250 diffusion steps for NWM and 50 flow-matching ODE steps for RAE-NWM. For fair comparison, all methods rank the same candidate trajectory sets generated by NoMaD (Sridhar et al., 2024). As shown in Table 12, UA-NWM consistently outperforms both baselines across the two datasets and all candidate-set sizes. Although UA-NWM is primarily designed for aerial navigation in 3D environments, these results demonstrate that its uncertainty-aware trajectory scoring mechanism also transfers effectively to 2D ground navigation, indicating broader applicability beyond the original setting.

A.7 Further Analysis of HEP

This section examines the foundations and empirical behavior of the proposed Hierarchical Error Projection (HEP). We first use a probabilistic view to explain why the unexplained residual serves as a trajectory–goal compatibility score, and then detail the geometry of the coarse-to-fine projection and how its training objective learns recurring residual directions under similar context–action conditions. Our purpose is to make the assumptions behind the HEP interpretation explicit, rather than to claim that HEP estimates a fully calibrated likelihood of future observations. We next evaluate whether the resulting score separates trajectory-compatible goals from hard-incompatible alternatives, and use paired per-case analysis to compare HEP with deterministic discrepancy scoring across offline ranking and planning. Together, these analyses connect the learned uncertainty directions to their role in trajectory selection.

A.7.1 Residual Energy as a Conditional Compatibility Score

For a fixed observation context $O_{t-C+1:t}$ and a candidate action sequence $A_{t:t+H-1}$, let μ be the deterministic DINO feature prediction and $x_g = \phi(o_g)$ be the DINO feature of the goal image. UA-NWM computes the discrepancy

$$e = \mathrm{norm}(x_g) - \mathrm{norm}(\mu). \tag{50}$$

To motivate the scoring principle, consider an idealized single-subspace model. Let E denote the residual induced by plausible futures under the same context and action. Suppose that most plausible variation around μ lies in an R-dimensional subspace $\mathcal{S} = \mathrm{span}(V)$, where the columns of V are orthonormal. A simple conditional residual model is

$$E = Va + \xi, \quad a \sim \mathcal{N}(0, \tau^2 I), \quad \xi \sim \mathcal{N}(0, \sigma^2 I). \tag{51}$$

Here, Va denotes uncertainty-explainable variation and ξ denotes residual variation outside the plausible subspace. The covariance is

$$\Sigma = \tau^2 V V^\top + \sigma^2 I. \tag{52}$$

For this Gaussian residual model, the negative log-likelihood of an observed residual contains the quadratic Mahalanobis term $e^\top \Sigma^{-1} e$; with fixed R, τ, and σ, the remaining log-determinant term is constant with respect to the observed residual. Within this idealized model, decomposing any residual as $e = e^{\parallel} + e^{\perp}$, with $e^{\parallel} \in \mathcal{S}$ and $e^{\perp} \perp \mathcal{S}$, gives

$$e^\top \Sigma^{-1} e = \frac{\|e^{\parallel}\|_2^2}{\tau^2 + \sigma^2} + \frac{\|e^{\perp}\|_2^2}{\sigma^2}. \tag{53}$$

The exact Gaussian score therefore penalizes both components, but with different weights. The HEP score follows the intuition of retaining the second term, corresponding to the high-anisotropy regime where plausible variation inside $\mathcal{S}$ is much larger than unexplained variation outside it, i.e., $\tau^2 \gg \sigma^2$. In this case, the penalty on $e^{\parallel}$ is strongly down-weighted and the dominant criterion becomes $\|e^{\perp}\|_2^2$. If τ^2 is not substantially larger than σ^2, a calibrated quadratic score would also retain a non-negligible penalty on $e^{\parallel}$; our score should then be viewed as a conservative compatibility approximation that treats variation inside $\mathcal{S}$ as acceptable for navigation. Importantly, this Gaussian decomposition is an idealized motivation. The implemented HEP applies sequential regularized fits across multiple scales, so its quantities $e^{\parallel}$ and $e^{\perp}$ are named by analogy and are not guaranteed to be mutually orthogonal. As detailed in Section A.7.2, the implemented score is the energy of the residual remaining after the hierarchy. A candidate is therefore penalized primarily when its goal discrepancy is not removed by the predicted uncertainty subspace, rather than simply when it differs from the deterministic mean prediction.

A.7.2 What the Hierarchical Projection Computes

The full feature discrepancy is a spatial field over N DINO patches. HEP constructs a multi-scale approximation of the plausible residual subspace using grid scales $\mathcal{G} = \{1, 2, 7, 14\}$. At scale g, the patch grid is partitioned into cells $\Omega_{g,c}$. Let $r_{g,i}$ denote the residual at patch i when entering scale g, initialized by $r_{1,i} = e_i$. The cell-average residual is

$$\bar{r}_{g,c} = \frac{1}{|\Omega_{g,c}|} \sum_{i \in \Omega_{g,c}} r_{g,i}. \tag{54}$$

The bar indicates that this is an average over all patches in the cell, not the residual of a single patch. For each cell, HEP predicts a basis $U_{g,c} = \{u_{g,c,1}, \ldots, u_{g,c,R}\}$. It then computes a ridge-regularized projection of $\bar{r}_{g,c}$ onto the span of this basis:

$$\alpha_{g,c} = \operatorname*{argmin}_{\alpha \in \mathbf{R}^R} \left\| \bar{r}_{g,c} - \sum_{r=1}^{R} \alpha_r u_{g,c,r} \right\|_2^2 + \lambda \|\alpha\|_2^2, \tag{55}$$

$$p_{g,c} = \sum_{r=1}^{R} \alpha_{g,c,r} u_{g,c,r}. \tag{56}$$

When $\lambda = 0$, $p_{g,c}$ is the ordinary least-squares projection of $\bar{r}_{g,c}$ onto $\mathrm{span}(U_{g,c})$. With $\lambda > 0$, it is a regularized projection that improves numerical stability and prevents unstable coefficients when the predicted basis vectors are correlated. The projected vector is then subtracted from every patch in the cell:

$$r_{g^+,i} = r_{g,i} - p_{g,c}, \quad i \in \Omega_{g,c}, \tag{57}$$

where g^+ denotes the next finer scale.

This operation has a clear geometric meaning. At each scale, HEP removes a low-rank, cell-wise constant residual component. Coarse scales explain broad spatially coherent deviations, while fine scales explain more localized deviations. The finest 14×14 scale reduces to patch-wise residual projection because each cell contains one DINO patch.

We distinguish HEP from a single exact orthogonal projection. Let $\mathcal{S}_g$ denote the space of residual fields that are constant within each cell of scale g and whose cell value lies in the corresponding local basis span, and let

$$\mathcal{S}_{\text{all}} = \mathcal{S}_1 + \mathcal{S}_2 + \mathcal{S}_7 + \mathcal{S}_{14}. \tag{58}$$

An exact orthogonal projection would solve one global least-squares problem,

$$p^\star = \operatorname*{argmin}_{p \in \mathcal{S}_{\text{all}}} \|e - p\|_2^2, \tag{59}$$

and would produce a residual $e - p^\star$ that is orthogonal to the whole space $\mathcal{S}_{\text{all}}$. This global solve is not what HEP implements.

Instead, HEP performs a one-pass coarse-to-fine residual decomposition. It first removes the component predicted at the coarsest scale, then applies the next scale to the remaining residual, and continues until the finest scale. This is greedy because each scale explains only the residual left by previous scales and the earlier components are not jointly re-optimized. If the scale-wise subspaces are mutually orthogonal, this sequential procedure is equivalent to the exact orthogonal projection onto their direct sum, since components from different scales do not interfere with each other. The implementation does not explicitly enforce this condition. Therefore, in the general case, $e^{\|}$ denotes the sum of the components removed across scales and $e^{\perp}$ denotes the remaining residual. The superscripts are retained as intuitive notation, but do not assert a strict parallel–orthogonal decomposition, and the score is not the exact Euclidean distance to a single global subspace. The remaining residual is the quantity used by UA-NWM for trajectory scoring.

A.7.3 Why the Training Objective Learns Plausible Residual Directions

During HEP training, the deterministic baseline is fixed and HEP is optimized with the normalized residual loss

$$\mathcal{L}_{\text{HEP}} = \frac{\sum_{i=1}^{N} \|e^{\perp}_{t+1,i}\|_2^2}{\sum_{i=1}^{N} \|e_{t+1,i}\|_2^2 + \epsilon}. \tag{60}$$

This objective encourages the predicted bases to explain residual directions that repeatedly occur under the same type of context and action. To connect the loss to subspace learning, consider a fixed scale-cell pair (g, c) and temporarily view its predicted basis span as a fixed R-dimensional subspace $\mathcal{S}$ and set aside ridge regularization. This idealized single-scale analysis uses the exact projection operator $P_{\mathcal{S}}$; it characterizes the learning incentive but is not an identity for the full multi-scale HEP procedure. For a single training sample, the cell-average residual $\bar{r}_{g,c}$ is decomposed into an explained part $P_{\mathcal{S}}\bar{r}_{g,c}$ and an unexplained part $(I - P_{\mathcal{S}})\bar{r}_{g,c}$. The loss reduces this unexplained component, while the denominator in $\mathcal{L}_{\text{HEP}}$ only rescales the sample contribution.

Over the whole training set, minimizing this residual term corresponds to minimizing the empirical average of the unexplained energy. In population form, this empirical average is written as

$$\mathbb{E}\left[\|(I - P_{\mathcal{S}})\bar{\rho}_{g,c}\|_2^2\right], \tag{61}$$

where $\bar{\rho}_{g,c}$ denotes the random variable corresponding to the cell-average residual $\bar{r}_{g,c}$ when a transition is sampled from the training distribution, and $P_{\mathcal{S}}$ is the projection onto $\mathcal{S}$. Minimizing this quantity is equivalent to maximizing the expected explained energy

$$\mathbb{E}\left[\|P_{\mathcal{S}}\bar{\rho}_{g,c}\|_2^2\right]. \tag{62}$$

Thus, for a fixed context-action condition, the optimal low-rank subspace contains the principal residual directions of plausible futures. HEP does not store a separate subspace for each condition. Instead, it learns a conditional mapping from the deterministic prediction and context features to the local basis $U_{g,c}$. Since the deterministic prediction μ depends on the candidate action sequence, different candidate trajectories can induce different predicted bases and therefore different uncertainty directions.

This argument also clarifies why the design does not simply memorize arbitrary training residuals. Each cell predicts only R basis directions, so it cannot explain all channel-space errors when $R \ll D$, where D is the DINO feature-channel dimension. At coarser scales, the same projected vector is subtracted from all patches in a cell, so only spatially coherent deviations can be explained at that scale. In addition, the deterministic baseline is frozen during HEP training, which prevents

Table 13: Controlled conditional-compatibility evaluation. OOD denotes a hard-incompatible proposal. Pair Acc. is the probability that an incompatible proposal receives a larger score than the compatible proposal. R@1 is the fraction of cases in which the compatible proposal has the lowest score among the $K + 1$ proposals.

K	Method	AUROC		AUPR-OOD	AUPR-ID	Pair Acc.		R@1	
		Estimate	95% CI			Estimate	95% CI	Estimate	95% CI
4	UA-NWM	**0.684**	$(0.673, 0.697)$	**0.892**	**0.334**	**87.75%**	$(86.08, 89.33)$	**79.0%**	$(76.5, 81.3)$
4	Deterministic Baseline	0.675	$(0.663, 0.687)$	0.888	0.325	85.53%	$(83.63, 87.28)$	75.0%	$(72.2, 77.6)$
8	UA-NWM	**0.666**	$(0.655, 0.678)$	**0.938**	**0.190**	**85.25%**	$(83.50, 86.98)$	**68.6%**	$(65.7, 71.5)$
8	Deterministic Baseline	0.657	$(0.646, 0.669)$	0.935	0.185	82.83%	$(80.99, 84.69)$	63.7%	$(60.7, 66.7)$
16	UA-NWM	**0.642**	$(0.632, 0.653)$	**0.964**	**0.096**	**82.02%**	$(80.28, 83.78)$	**56.3%**	$(53.1, 59.4)$
16	Deterministic Baseline	0.634	$(0.623, 0.645)$	0.963	0.093	79.56%	$(77.64, 81.48)$	51.3%	$(48.2, 54.3)$

Table 14: Paired comparison with the shared-backbone deterministic scorer. Δ is UA-NWM minus deterministic; negative favors UA-NWM.

Task	Metric	UA-NWM	Deterministic Baseline	Δ (95% CI)
Ranking (8 Candidates)	ATE $\downarrow$	1.256	1.391	$\mathbf{-0.135}$ $(-0.158, -0.112)$
	RPE $\downarrow$	0.347	0.380	$\mathbf{-0.033}$ $(-0.039, -0.028)$
Ranking (16 Candidates)	ATE $\downarrow$	1.174	1.325	$\mathbf{-0.151}$ $(-0.175, -0.129)$
	RPE $\downarrow$	0.326	0.364	$\mathbf{-0.038}$ $(-0.044, -0.032)$
Ranking (32 Candidates)	ATE $\downarrow$	1.091	1.267	$\mathbf{-0.176}$ $(-0.199, -0.153)$
	RPE $\downarrow$	0.304	0.348	$\mathbf{-0.043}$ $(-0.049, -0.038)$
Planning	ATE $\downarrow$	1.222	1.262	$\mathbf{-0.040}$ $(-0.050, -0.030)$
	RPE $\downarrow$	0.334	0.346	$\mathbf{-0.012}$ $(-0.015, -0.009)$

HEP from changing the mean prediction μ to absorb residual errors. These restrictions make the easiest residuals to explain those that appear consistently under similar context-action conditions. Therefore, the HEP loss provides a principled surrogate for learning conditional plausible-variation directions, while the final navigation score uses the remaining unexplained residual to distinguish goal-compatible and goal-incompatible candidate trajectories.

A.7.4 Controlled Evaluation of Conditional Goal Compatibility

We quantify how well the HEP residual separates goals compatible with a candidate trajectory from incompatible goals using all 1,000 AirGoal-10k test cases, each with 32 NoMaD proposals. Within each case, the proposal with minimum full-trajectory ATE is labeled compatible, and the K proposals with largest ATE are labeled hard-incompatible, for $K \in \{4, 8, 16\}$. Confidence intervals use 2,000 case-level bootstrap resamples. Across all three difficulty settings, Table 13 shows that UA-NWM consistently improves discrimination, pairwise ordering, and retrieval over the deterministic baseline. Because the two methods share the predictor and visual features, the gain isolates the contribution of the HEP.

A.7.5 Paired Statistical Analysis of HEP

We further compare UA-NWM with the deterministic baseline using paired per-case outcomes. For each test case, we compute the difference between the outcomes of the two methods and use 10,000 paired case-level bootstrap resamples to estimate the confidence interval of the mean difference.

Table 14 reports the results for both ATE and RPE across the three offline-ranking settings and offline planning. UA-NWM obtains lower mean errors in every comparison, and all reported differences are negative. Moreover, none of the eight 95% confidence intervals includes zero. The consistency across candidate-set sizes and planning indicates that HEP improves trajectory scoring beyond the shared deterministic prediction.

A.8 Further Discussions and Future Work

In Figure 5, we visualize alternative plausible states sampled from the predicted uncertainty subspace. Here, we detail the sampling procedure and further discuss the capability and role of UA-NWM as a world model.

As illustrated in Figure 11(a), UA-NWM first predicts the deterministic future state μ and the uncertainty subspace $\mathcal{S}$. We then sample different points within $\mathcal{S}$ to obtain x_g^P and other plausible future states.

The reconstructed state $x_g^P = \mu + e^{\parallel} = x_g - e^{\perp}$ is obtained by adding to μ the components successively removed by HEP. Compared with the discrepancy between μ and x_g, the decoded x_g^P typically exhibits a substantially smaller gap from x_g. It also tends to preserve a plausible appearance because it is derived from μ through hierarchical error projection, which is trained to explain residuals along plausible uncertainty directions.

Nevertheless, the uncertainty subspace remains only a low-dimensional, local approximation to the conditional future-state distribution. Consequently, not every point in $\mathcal{S}$ lies in a high-density region of the true distribution, and random sampling in it does not always yield high-quality results. As shown in Figure 11(b), these samples exhibit meaningful diversity from the mean prediction μ and generally preserve reasonable global layouts, but their spatial coherence is limited. In particular, different regions of the same building may exhibit inconsistent colors, making the image resemble a shuffled jigsaw puzzle. This likely arises because, although the hierarchical projection captures some spatial correlations across image regions, the learned uncertainty subspace remains insufficient to enforce fully coherent variations among correlated patches. When multiple building colors are plausible, independent sampling within $\mathcal{S}$ may assign different colors to correlated patches rather than selecting a coherent appearance for the entire structure.

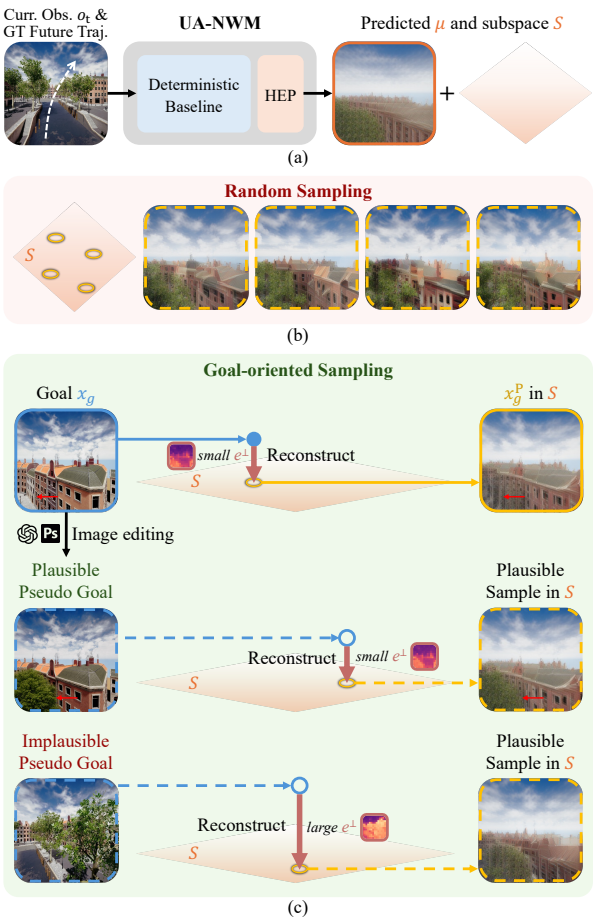


Figure 11: Different strategies of sampling plausible states on predicted subspace.

To obtain more interpretable samples, we introduce a goal-oriented sampling strategy. Inspired by the construction of x_g^P, we generate alternative plausible samples from modified goal images. As shown in Figure 11(c), we first use image-editing tools to construct a visually distinct yet contextually plausible pseudo-goal image from the original goal image x_g. If the pseudo-goal is plausible under the given initial observation and action sequence, it should lie near a high-density region of the same conditional future-state distribution, and processing its discrepancy through HEP should produce a reconstructed state that remains close to the pseudo-goal while differing in appearance from x_g^P.

Conversely, if the pseudo-goal is incompatible with the initial observation or action sequence, the components removed by HEP are insufficient to reproduce it. In the last example of Figure 11(c), the pseudo-goal is captured from another 3D position in the simulation environment and corresponds

to the endpoint of a different trajectory. It is therefore incompatible with the conditional future-state distribution induced by the original trajectory. The reconstructed state preserves the overall layout associated with the original trajectory rather than reproducing the pseudo-goal, while the large remaining residual allows UA-NWM to identify this incompatibility.

These observations suggest that UA-NWM is effective as a navigation discriminator and also shows promise as an efficient stochastic generator. Through hierarchical error projection, it can assess whether a candidate trajectory is compatible with the goal image while accounting for future-state uncertainty, making it well suited to image-goal navigation. However, fully realizing future-distribution prediction for efficient stochastic generation requires a more expressive uncertainty representation. Future work may explore more expressive uncertainty representations that capture spatial coherence across patches and temporal coherence across frames more precisely, enabling stochastic future generation by predicting a future-state distribution in a single forward pass and sampling diverse futures from it without running the full model again for each sample.

A.9 Additional Visualization Results

A.9.1 Hierarchical Error Projection

Visualizations of the hierarchical error projection process are shown in Figure 12. As the projection proceeds, the unexplained residual progressively decreases. Starting from μ, we gradually add the explained component $e^{\parallel}$ on it, ultimately obtaining x_g^P that closely resembles x_g.

A.9.2 Uncertainty Subspace

Additional visualizations of samples from the predicted uncertainty subspace are shown in Figure 13. For each case, besides x_g^P, we identify two alternative plausible samples in $\mathcal{S}$, demonstrating that the predicted subspace captures meaningful future-state diversity, primarily arising from occlusion-induced ambiguity or long-horizon drift.

A.9.3 Offline Trajectory Ranking

Qualitative comparisons on the offline trajectory ranking task are shown in Figure 14. When ranking 32 candidates, NWM (Bar et al., 2025) is susceptible to error accumulation and sampling variability, which may cause it to favor inferior trajectories. In contrast, UA-NWM leverages HEP to score trajectories using the unexplained residual, yielding more robust rankings that favor candidates closer to the ground-truth trajectory under future-state uncertainty.

A.9.4 Offline Standalone Planning

Qualitative comparisons on the offline standalone planning task are shown in Figures 15 and 16. Under the same CEM planning framework, UA-NWM effectively identifies the direction toward the goal through iterative sampling, prediction, and optimization, whereas NWM (Bar et al., 2025) often fails to recover the correct path.

A.9.5 Online Simulation

Qualitative results from the online simulation experiments are shown in Figure 17. Given a goal image with an unknown target distance, UA-NWM performs multi-step CEM planning and progressively approaches the target location through closed-loop planning.

A.9.6 Real-world UAV Navigation

Additional qualitative results from the real-world UAV experiments are shown in Figure 18. Across varying weather conditions, including sunny, cloudy, and post-rain conditions, and different times of day, including morning, afternoon, and dusk, UA-NWM demonstrates strong sim-to-real transfer and robust generalization in real-world scenes.

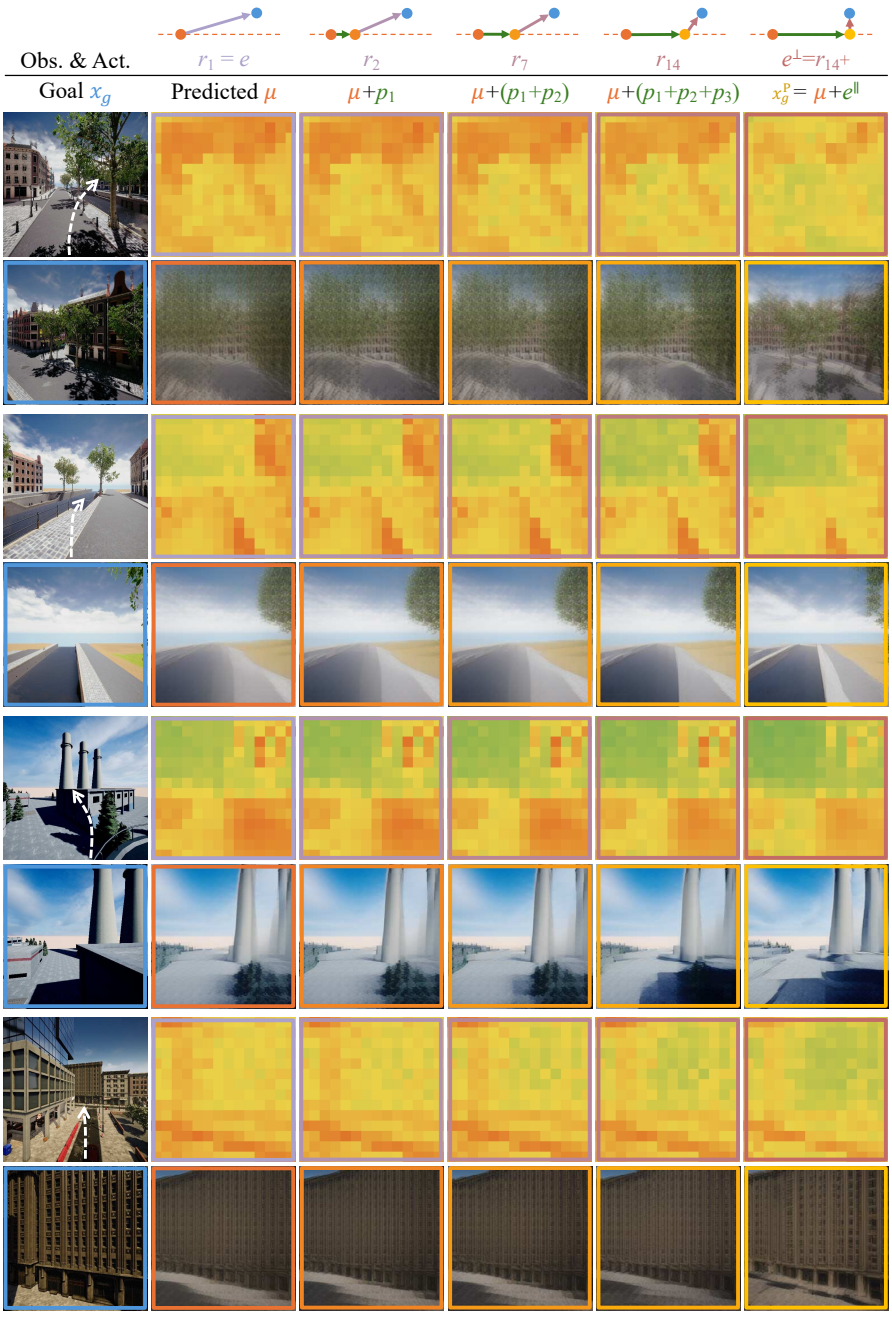


Figure 12: Visualizations of hierarchical error projection process.



Figure 13: Additional visualizations of samples from the predicted uncertainty subspace.

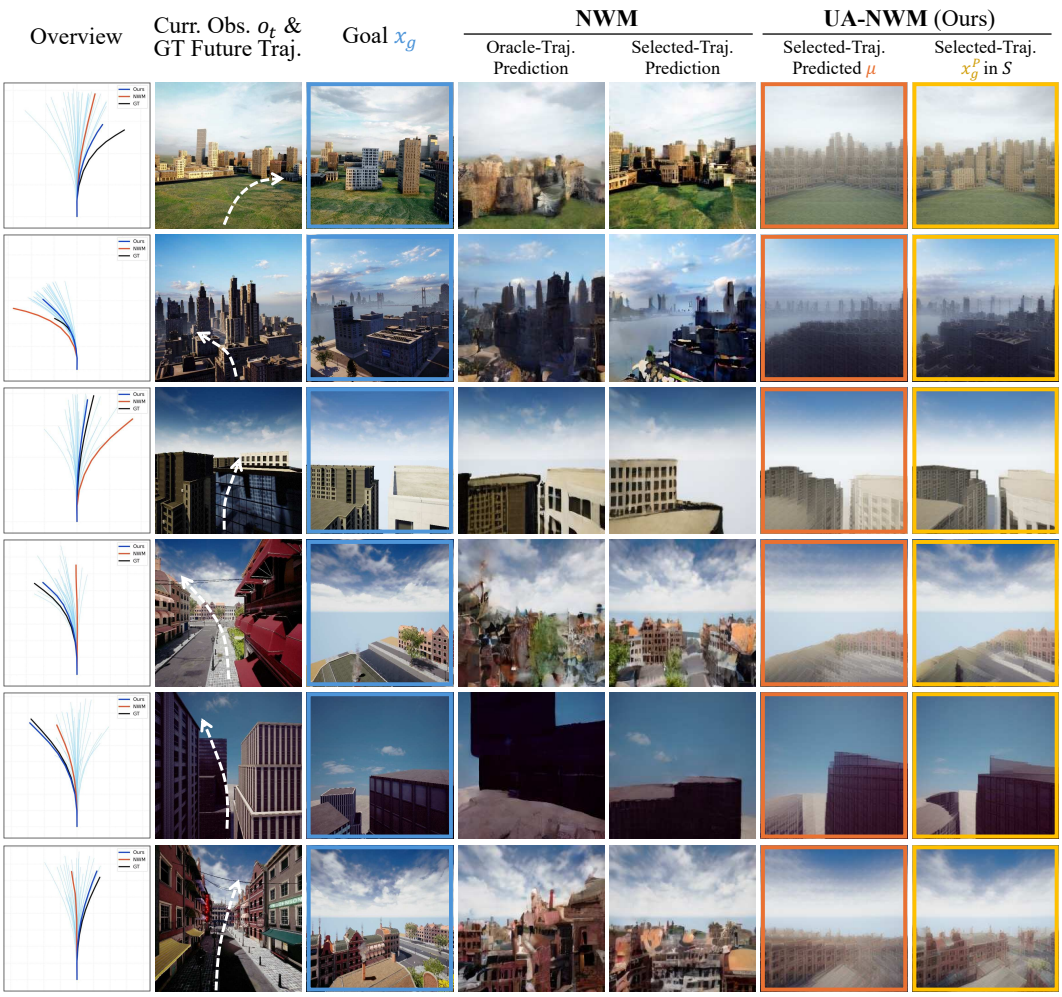


Figure 14: Qualitative comparisons on the trajectory ranking task.



Figure 15: Qualitative comparisons on the standalone planning task.

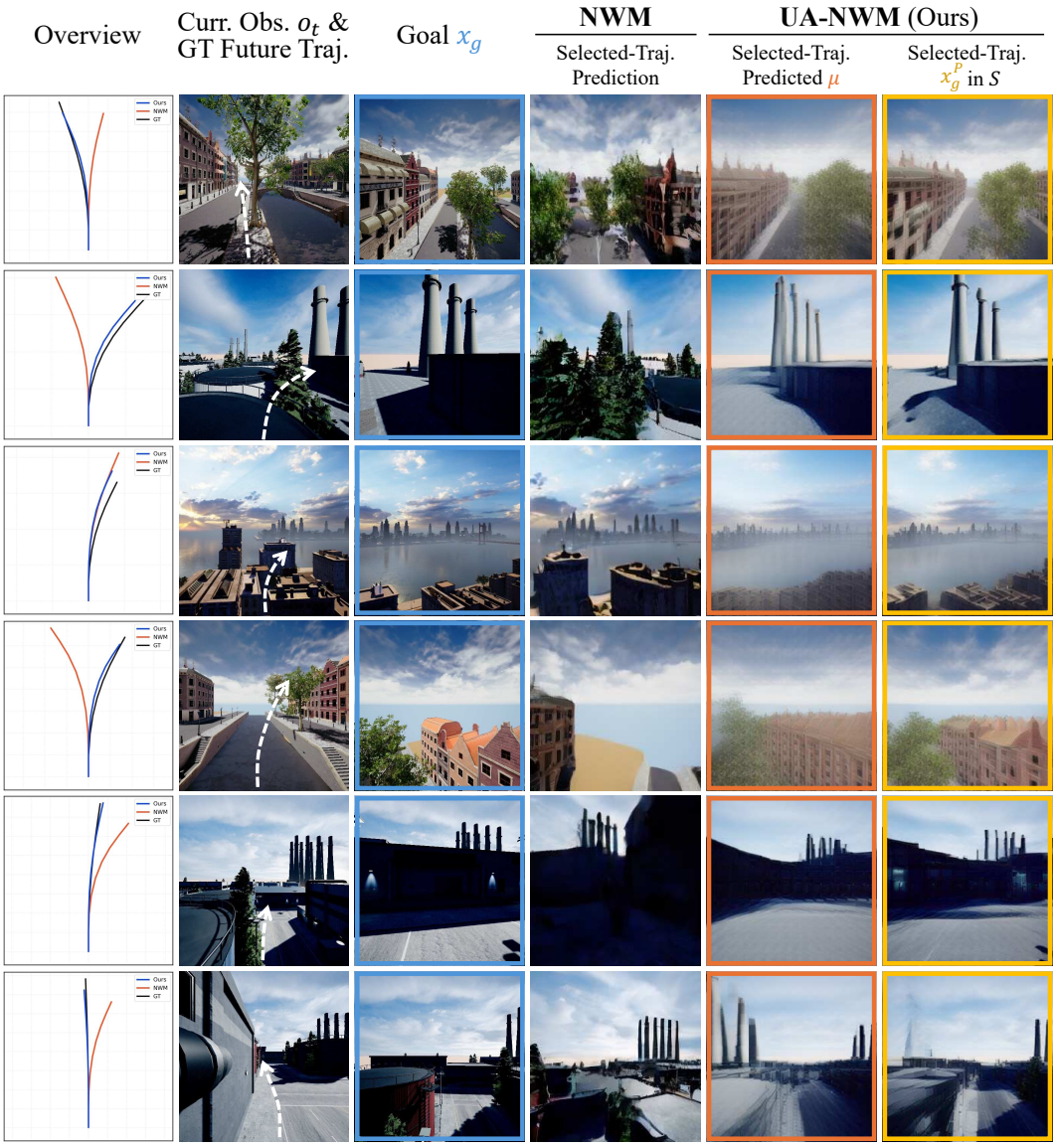


Figure 16: Qualitative comparisons on the standalone planning task.

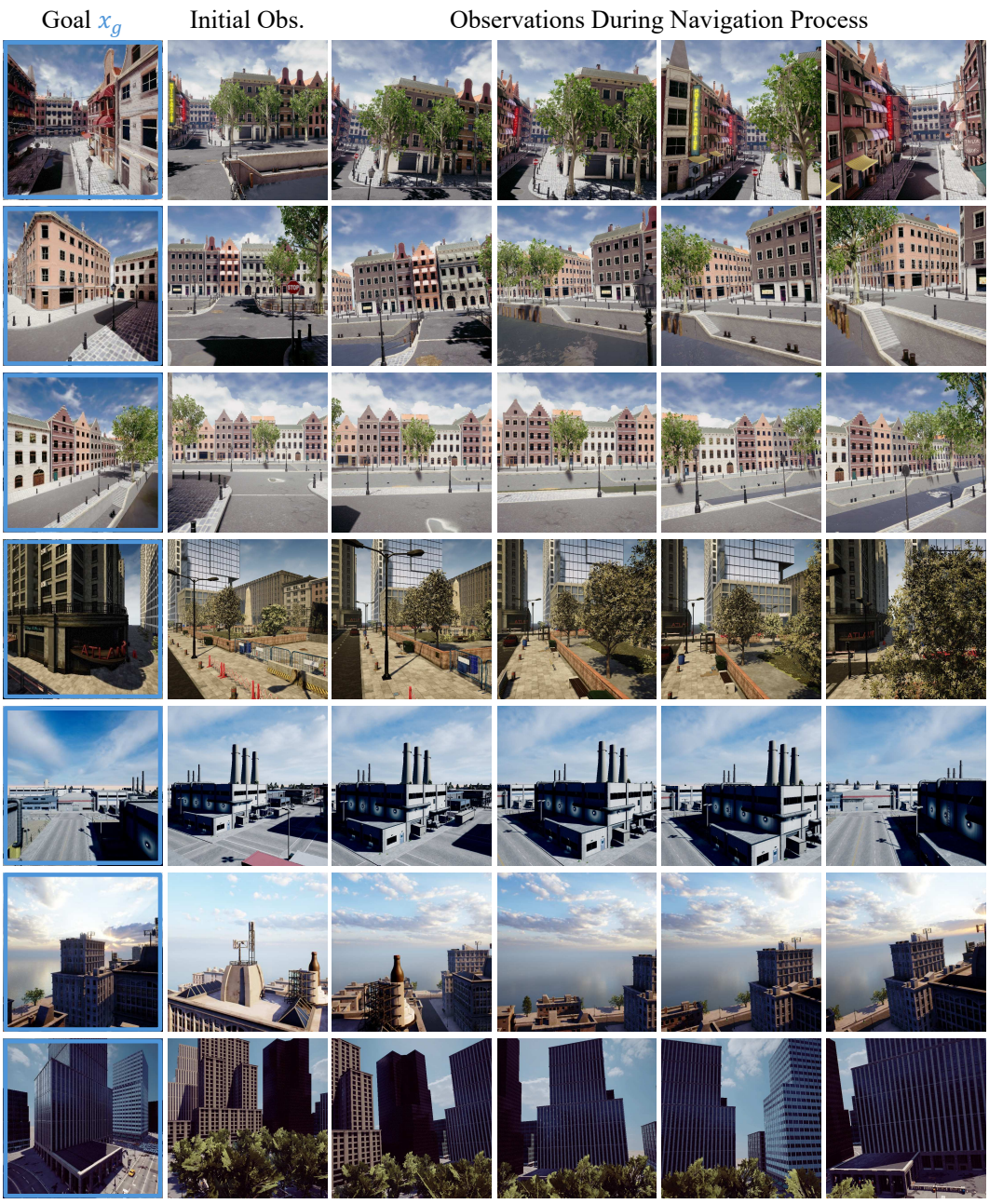


Figure 17: Qualitative results of online closed-loop navigation experiments.

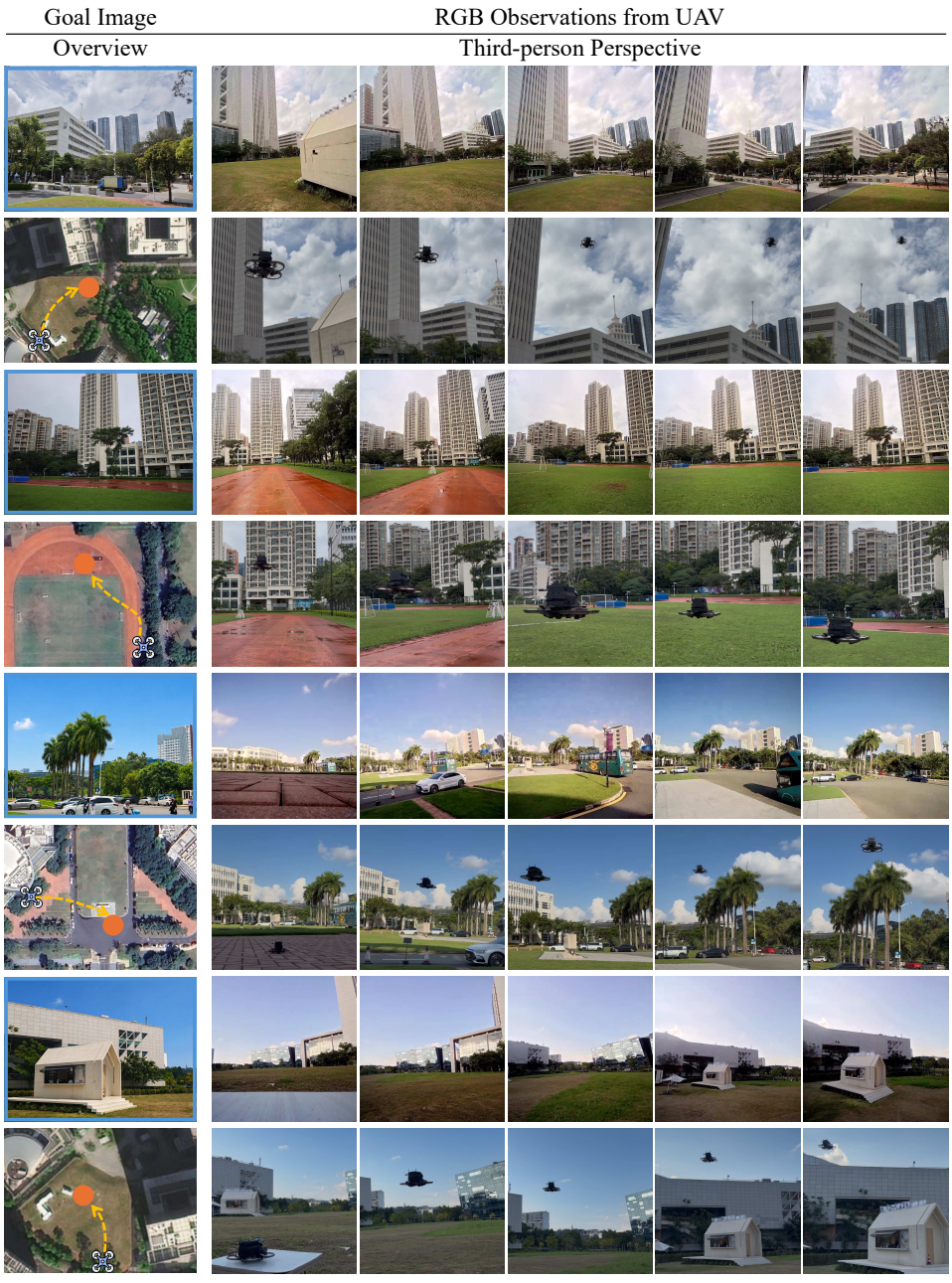


Figure 18: Additional qualitative results of real-world UAV experiments. Identifying information has been blurred to preserve anonymity during double-blind review.