WorldVLN: Autoregressive World Action Model for Aerial Vision-Language Navigation

Baining Zhao[1]*, **Jiacheng Xu**[2]*, **Weicheng Feng**[2]*, **Xin Zhang**[3]*, **Zhaolu Wang**[4]*, **Haoyang Wang**[1], **Shilong Ji**[1], **Ziyou Wang**[1], **Jianjie Fang**[1], **Zhiheng Zheng**[1], **Weichen Zhang**[1], **Yu Shang**[1], **Wei Wu**[3], **Chen Gao**[1]†, **Xinlei Chen**[1]†, **Yong Li**[1]
[1]Tsinghua University, [2]Shandong University,
[3]Manifold AI, [4]Zhongguancun Academy
zbn22@mails.tsinghua.edu.cn, chgao96@gmail.com,
chen.xinlei@sz.tsinghua.edu.cn, liyong07@tsinghua.edu.cn

Abstract

Aerial vision-language navigation (VLN) requires agents to follow natural-language instructions through closed-loop perception and action in 3D environments. We argue that aerial VLN can be formulated as a prediction-driven world-action problem: the agent should anticipate latent world evolution and act according to the predicted consequences. To this end, we propose WorldVLN, a streamlined, natively autoregressive world action model for aerial VLN. Unlike full-sequence video-generation world models that generate an entire visual clip, WorldVLN adapts a latent autoregressive video backbone to predict short-horizon world-state transitions and directly decodes them into executable waypoint actions. After each action segment is executed, newly received observations are encoded back into the autoregressive context, enabling closed-loop world-action prediction. We further introduce a two-stage training framework that first grounds the video prior in instruction-conditioned navigation dynamics and then develops Action-aware GRPO, a reinforcement learning method tailored to autoregressive WAMs, to optimize waypoint decisions through their downstream rollout consequences. On public outdoor and indoor benchmarks, WorldVLN improves average success rates over the strongest baselines by more than 14 percentage points on both benchmarks, with a larger gain on the hardest indoor split. It further transfers zero-shot to real drone deployment, suggesting that the proposed WorldVLN offers a promising route for spatial action tasks. Demos and code are available at https://embodiedcity.github.io/WorldVLN/.

1 Introduction

Vision-Language Navigation (VLN) is one of the core tasks of spatial intelligence, where an agent follows human instructions and autonomously moves through a 3D environment [1, 23, 46, 55]. The agent must understand high-level language, perceive partial egocentric observations, and progressively generate low-level actions in a closed-loop manner as new observations become available [23, 35, 37], making generalizable VLN agents difficult to build. The rapid progress of foundation models, especially LLMs and VLMs, has created new opportunities to transfer general-purpose capabilities to embodied navigation [56, 52, 32]. Following the Bitter Lesson [38], Vision–Language–Action (VLA) models extend vision-language models with action outputs and directly map observations and instructions to control commands [3, 21, 50]. However, VLA models still suffer from limited

*All authors contributed equally to this research.
†Corresponding authors.

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

generalization in embodied navigation, because their web-scale visual-linguistic priors are well suited for recognizing objects and parsing instructions, but not for modeling how the world evolves under the agent's own actions. As a result, they remain limited in capturing the temporal, geometric, and causal structure required for embodied action generation, treating embodied behavior as a conditional **mapping** from instruction and observation to action.

Actually, biological spatial intelligence [11, 9, 12] suggests that navigation is inherently anticipatory: humans implicitly predict the state consequences of their own movements and select actions that are expected to bring the resulting state closer to the intended goal. Recent advances in visual foundation models, especially video generation models [48, 39], have revealed the emergence of powerful predictive capabilities from large-scale visual-temporal pretraining [20, 22]. Extending this direction, video-based world models learn how visual scenes evolve under action-conditioned dynamics and thereby acquire rich spatiotemporal priors over motion, viewpoint transitions, and physical evolution—precisely the structure that VLM-based VLA models lack. This observation reveals a different formulation of VLN as a **prediction** problem: given observations and instructions, the agent predicts how the world will evolve under candidate movements and selects the action whose anticipated consequences best satisfy the instructed goal.

However, realizing spatial action with existing video generation models remains nontrivial. A direct approach [5, 10] is to condition a video generation model on the VLN instruction and current observation, synthesize future visual observations, and then recover actions through visual odometry. Yet this pipeline exposes fundamental mismatches between video generation and embodied navigation, both in model structure and in the learning objective that shapes the underlying representations:

- **Model Structure:** Most video generation backbones [44, 6] generate an entire clip in a bidirectional manner, whereas embodied navigation requires a causal observe–act–update loop grounded in past and current observations. This mismatch is especially critical in aerial VLN, where large viewpoint changes and accumulated state errors require persistent memory and closed-loop correction [28, 54].
- **Learning Objective:** Generic video generation models are optimized for visually plausible synthesis [17, 16], whereas VLN requires action-aware consequence modeling: the learned representations must not only predict how observations evolve, but also encode which state transitions are geometrically consistent, action-decodable, and beneficial for reaching the instructed goal.

To address these challenges, we propose **WorldVLN**, an autoregressive world action model (WAM) for aerial VLN, together with a two-stage training framework that aligns a video-generation backbone with world-action dynamics. **First**, WorldVLN repurposes a pre-trained video latent autoregressive transformer for closed-loop navigation. The backbone predicts short-horizon latent world transitions from the instruction and observation history, decodes them into waypoint actions via a designed action decoder, and feeds newly observed states back into the autoregressive context after execution. **Then**, we train WorldVLN with a two-stage framework. Stage 1 uses supervised training to ground the video prior in instruction-conditioned navigation dynamics and train the action decoder to recover expert waypoint actions from latent world transitions. Stage 2 introduces **Action-aware Group Relative Policy Optimization (GRPO)**, which performs online autoregressive rollouts and optimizes segment-level action decisions with trajectory, task, and reference rewards. A temporal decay weighting further emphasizes early decisions, encouraging the model to account for how current actions influence downstream observations, future actions, and final navigation success. **Finally**, experiments on public indoor and outdoor UAV benchmarks show that WorldVLN significantly outperforms VLA baselines. We further explore three questions—whether WAM learns more effectively than VLA, why autoregressive prediction is necessary, and what Action-aware GRPO contributes—highlighting the effectiveness of the proposed architecture and training algorithm. WorldVLN also shows zero-shot transfer to real UAV deployment. Our main contributions are summarized as follows:

- To our knowledge, we propose a **streamlined, natively autoregressive** world-action model for aerial VLN, which temporally predicts latent world representations, directly decodes low-level navigation actions, and closes the loop by grounding subsequent decisions in newly received visual observations.
- We introduce Action-aware GRPO, a **reinforcement learning post-training method** designed for autoregressive WAM policies in aerial navigation. After supervised navigation grounding, our Action-aware GRPO further aligns latent world-action representations with navigation outcomes.

- We achieve state-of-the-art results on both outdoor and indoor challenging aerial VLN benchmarks, and demonstrate zero-shot generalization on a real-world drone platform.

2 Related work

Vision-language-action models. VLA models extend pretrained vision-language models with action heads, enabling end-to-end mapping from language instructions and visual observations to executable actions [21, 50, 42]. This paradigm has been widely explored in embodied control, including robotic manipulation [3, 19] and navigation [14, 47], by transferring semantic priors from large-scale vision-language pretraining. However, their generalization in embodied navigation remains limited, because VLM-based VLAs primarily inherit priors for object recognition, instruction parsing, and scene understanding, but do not explicitly model action-conditioned world dynamics.

World action models. Video generation foundation models [48, 39, 44] provide strong visual-temporal priors over motion, viewpoint changes, and scene evolution, making them promising world modeling backbones [51, 33]. However, they are primarily optimized for realistic future synthesis rather than goal-directed action generation. Recent navigation methods often use world models in an imagine-and-rank manner, where multiple candidate routes are visually rolled out and then selected according to predicted outcomes [2, 53]. Although effective, this paradigm is indirect and computationally expensive. WAMs offer a more integrated alternative by coupling latent world prediction with action generation, either by recovering actions from predicted futures or directly decoding actions from world representations [22, 31]. Autoregressive WAMs [25, 49] further support temporal memory and closed-loop feedback, but remain underexplored for spatial navigation, especially aerial VLN, where large viewpoint changes, continuous 3D motion, and accumulated state errors make autoregressive updating particularly important. Moreover, existing autoregressive WAMs often adapt bidirectional diffusion backbones via teacher- or self-forcing, incurring high computational cost and hindering direct optimization under the native observe–act–update interface.

Post-training methods. Post-training adapts pretrained foundation models to downstream task objectives. At the data level, recent studies [26, 43] develop automatic annotation pipelines for navigation instructions, leveraging Internet videos and synthetic data generation to alleviate the scarcity of annotated VLN data. Current VLA and WAM policies are still largely trained by supervised fine-tuning on expert demonstrations, with WAMs sometimes using imagined observations or video prediction as additional supervision. Such imitation-based training fits the demonstration distribution but remains vulnerable to covariate shift and accumulated errors [34, 30]. Reinforcement learning can directly optimize task rewards beyond demonstration likelihood [24, 4, 36], but reinforcement learning (RL) for autoregressive WAMs [8] remains underexplored. Applying RL to autoregressive WAMs raises new challenges, including the mismatch between visually plausible world generation and action-outcome optimization, as well as credit assignment in multi-step closed-loop rollouts [15, 45].

3 Problem formulation

We formulate aerial VLN as a partially observable sequential decision-making problem. Given a natural-language instruction ℓ at the starting position, the agent executes a sequence of actions to progressively complete the instruction in a 3D environment. Let π_θ denote the navigation policy. At step t, the agent receives an egocentric observation o_t and predicts a waypoint action a_t conditioned on the instruction, the current and historical observations, and the executed action history:

$$a_t \sim \pi_\theta(\cdot \mid o_{\le t}, a_{<t}, \ell), \qquad a_t = (\Delta x_t, \Delta y_t, \Delta z_t, \Delta\psi_t) \in \mathbb{R}^4, \tag{1}$$

where $(\Delta x_t, \Delta y_t, \Delta z_t)$ represents the relative 3D translation and $\Delta\psi_t$ represents the relative yaw change. Executing a_t updates the agent pose $q_t = (x_t, y_t, z_t, \psi_t)$ and induces a new observation:

$$q_{t+1} = q_t \oplus a_t, \qquad o_{t+1} = \Omega(q_{t+1}), \tag{2}$$

where $\oplus$ applies the relative waypoint displacement to the current pose, and Ω maps the updated pose to the corresponding egocentric visual observation in the 3D environment. The agent repeats this observe–act process until it predicts a stop action or reaches the maximum horizon. The navigation is considered successful if the final position (x_T, y_T, z_T) is within a threshold ϵ of the ground-truth target position $(x^\star, y^\star, z^\star)$: $\|(x_T, y_T, z_T) - (x^\star, y^\star, z^\star)\|_2 < \epsilon$.

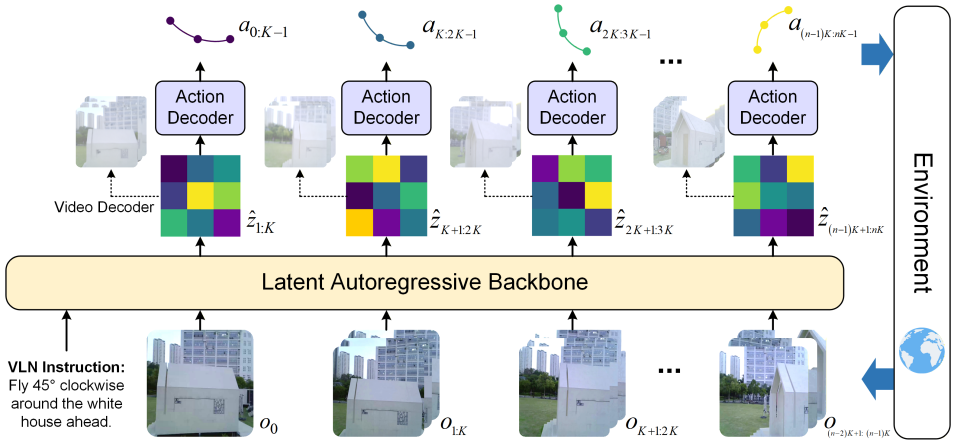


Figure 1: **WorldVLN architecture.** The model predicts short-horizon latent world transitions from the instruction and observation history, decodes them into waypoint actions, and updates the **autoregressive** context with newly observed states after execution. See Appendix A.3 for details.

4 Method

We first introduce the WorldVLN architecture for autoregressive world-action prediction and then describe the two-stage training framework that combines supervised grounding with Action-aware GRPO.

4.1 Model architecture

As shown in Figure 1, WorldVLN adopts a pre-trained latent autoregressive video transformer as the world backbone to capture the temporal evolution of the agent's embodied state in latent space. Rather than treating the predicted latent as a video segment to be rendered, WorldVLN reinterprets it as a short-horizon world-state transition induced by the agent's movement, which provides the basis for action prediction.

The backbone consists of a text encoder ψ, a video VAE encoder $\mathcal{E}_{\text{vid}}$, and a latent autoregressive Transformer p_θ. Let $e_\ell = \psi(\ell)$ be the encoded instruction, let K denote the prediction horizon, and let $z_{\le t}$ denote the latent context encoded from real egocentric observations up to time t. Following the temporal autoregressive structure of the video backbone, the next latent world segment is predicted as

$$\hat{z}_{t+1:t+K} \sim p_\theta\left(\cdot \mid e_\ell, z_{\le t}\right). \tag{3}$$

In the original video generation setting, $\hat{z}_{t+1:t+K}$ would be decoded into future frames and then used as part of the context for subsequent generation. In WorldVLN, we instead feed $\hat{z}_{t+1:t+K}$ to the action decoder D_ϕ:

$$a_{t:t+K-1} = D_\phi\left(\hat{z}_{t+1:t+K}\right). \tag{4}$$

After executing $a_{t:t+K-1}$, the agent receives real egocentric observations $o_{t+1:t+K}$, which are encoded into real latents:

$$z_{t+1:t+K} = \mathcal{E}_{\text{vid}}\left(o_{t+1:t+K}\right). \tag{5}$$

Instead of continuing generation with the model-predicted latent $\hat{z}_{t+1:t+K}$, WorldVLN replaces it with the real latent $z_{t+1:t+K}$ in the autoregressive context. The resulting closed-loop rollout is:

$$(e_\ell, z_0) \to \hat{z}_{1:K} \to a_{0:K-1} \to o_{1:K} \to z_{1:K} \to \hat{z}_{K+1:2K} \to \cdots. \tag{6}$$

Thus, each generated latent segment is used to decode a waypoint action sequence, while subsequent autoregressive prediction is grounded in the real latent encoded from the actual observation segment.

4.2 Training framework

We train the autoregressive WAM in two stages, as shown in Figure 2. Stage 1 uses supervised training to ground the video prior in instruction-conditioned navigation dynamics and make latent world representations action-decodable. Stage 2 introduces **Action-aware GRPO**, which uses online rollout rewards to optimize waypoint decisions according to their navigation consequences.

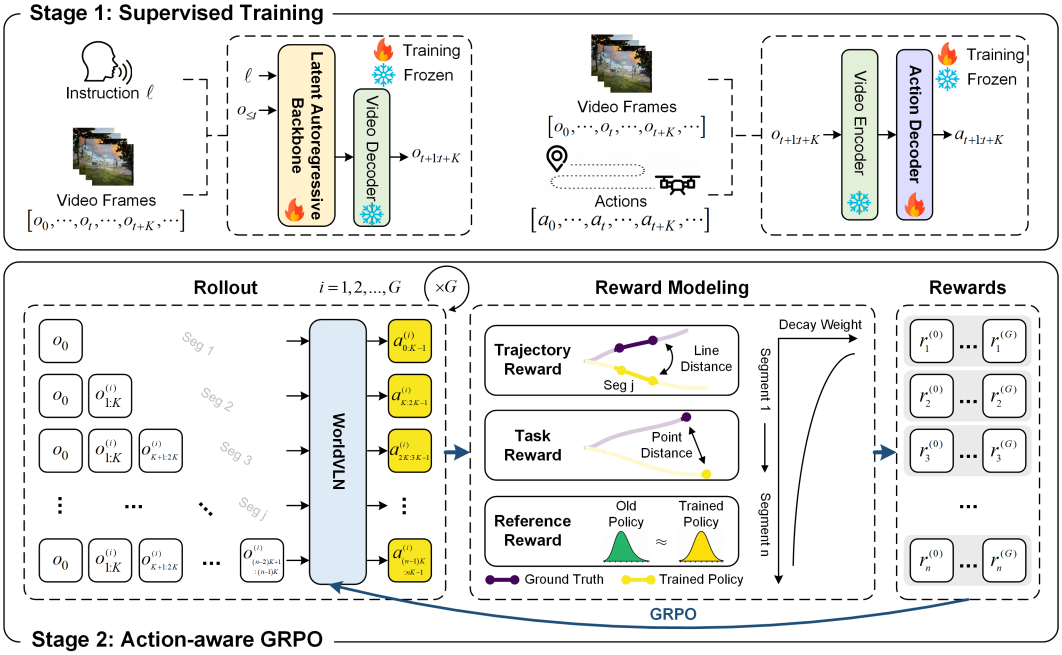


Figure 2: **Training framework**. Stage 1 supervises the latent autoregressive backbone with instruction-video pairs and the action decoder with video-trajectory pairs. Stage 2 samples multiple rollouts, computes segment rewards from trajectory accuracy, task progress, and reference-policy regularization, and uses temporally weighted advantages to update WorldVLN through **Action-aware GRPO**.

4.2.1 Stage 1: Supervised training

We train the backbone and the action decoder with supervised objectives, respectively. For the latent autoregressive backbone, we place the model back into its original autoregressive video-generation formulation, but use paired navigation instructions and egocentric navigation videos as training data. We encode the instruction ℓ as $e_\ell = \psi(\ell)$, and divide the corresponding video into $n+1$ observation segments $\{o_0, o_{1:K}, \ldots, o_{(n-1)K+1:nK}\}$. Each segment is encoded into a latent representation $z_{t+1:t+K} = \mathcal{E}_{\text{vid}}\left(o_{t+1:t+K}\right), \quad \text{for } t = 0, K, \ldots, (n-1)K$. Following the temporal autoregressive objective of the backbone, we train it to predict each ground-truth future latent segment from the instruction and previous ground-truth latent context:

$$\mathcal{L}_{\text{wm}} = -\sum \log p_\theta\left(z_{t+1:t+K} \mid e_\ell, z_{\le t}\right). \tag{7}$$

For the action decoder, we use paired navigation videos and trajectories as training data. Each video and trajectory are similarly divided into segments $\{o_{t+1:t+K}, a^*_{t:t+K-1}\}$. We encode each video segment as $z_{t+1:t+K}$ and train the action decoder to recover the expert action. This matches the final inference interface, where the decoder receives a latent world transition and outputs executable waypoint actions. To accelerate convergence, we initialize the decoder with features from the video decoder and a learning-based visual odometry backbone, as both provide useful priors for mapping visual state transitions to camera-pose motion. The action decoder is optimized by

$$\mathcal{L}_{\text{act}} = \sum \left\| D_\phi\left(\mathcal{E}_{\text{vid}}\left(o_{t+1:t+K}\right)\right) - a^*_{t:t+K-1} \right\|. \tag{8}$$

4.2.2 Stage 2: Action-aware GRPO

To further align the autoregressive WAM with navigation outcomes, we introduce action-aware GRPO. During training, the model performs online autoregressive rollouts in the simulator, following the same observe–act process used at inference time. Given an instruction and the initial observation,

the model predicts a latent segment, decodes waypoint actions, executes them in the environment, receives new observations, and repeats this process until the end of the navigation case.

For each navigation case, a group of G online rollouts is sampled from the current policy and each rollout contains n autoregressive decision segments. Suppose the j-th action segment in the i-th rollout is denoted by $a^{(i)}_{(j-1)K:jK-1}$. We assign reward to it by:

$$r^{(i)}_j = \gamma^{j-1}\left(\lambda_{\text{traj}} r^{(i)}_{\text{traj},j} + \lambda_{\text{task}} r^{(i)}_{\text{task},j} + \lambda_{\text{ref}} r^{(i)}_{\text{ref},j}\right). \tag{9}$$

Trajectory reward provides local geometric supervision by measuring how closely the predicted action a follows the expert action $a^\star$. Since each segment consists of multiple waypoints, we compute the trajectory distance and convert it into a reward:

$$r^{(i)}_{\text{traj},j} = \frac{1}{1 + \left\| a^{(i)}_{(j-1)K:jK-1} - a^{\star(i)}_{(j-1)K:jK-1} \right\|}. \tag{10}$$

Task reward provides global outcome evaluation by evaluating how the autoregressive rollout induced by local actions affects final goal reaching. We compute the terminal distance between the rollout endpoint and the ground-truth target, and assign a higher reward to goal-reaching outcomes:

$$r^{(i)}_{\text{task},j} = \frac{1}{1 + \left\| (x^{(i)}_T, y^{(i)}_T, z^{(i)}_T) - (x^{\star(i)}, y^{\star(i)}, z^{\star(i)}) \right\|_2}. \tag{11}$$

Reference reward regularizes the updated policy toward the reference policy to prevent excessive policy drift. This term helps preserve the implicit imagination capability learned during supervised world-action training, preventing the latent world prediction from drifting too far away from the original navigation dynamics prior. We evaluate the probability of the sampled segment action under the reference policy π_{ref}:

$$r^{(i)}_{\text{ref},j} = \log \pi_{\text{ref}}\left(a^{(i)}_{(j-1)K:jK-1} \mid o_{\le t}, a^{(i)}_{<(j-1)K}, \ell\right). \tag{12}$$

Temporal weighting. For each decision segment j, we compute the mean μ_j and standard deviation σ_j of its rewards across the G rollouts. The reward-level factor γ^{j-1} is shared across the rollouts at segment j and largely cancels under group normalization. We therefore also apply temporal decay to the normalized advantage, yielding $A^{(i)}_j = \gamma^{j-1}(r^{(i)}_j - \mu_j)/(\sigma_j + \epsilon)$ with $0 < \gamma < 1$. This weighting gives earlier decisions more influence in the clipped GRPO objective because they affect more subsequent observations and actions:

$$\mathcal{J}_{\text{GRPO}} = \mathbb{E}_{i,j}\left[\min\left(\rho^{(i)}_j A^{(i)}_j, \text{clip}\left(\rho^{(i)}_j, 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}}\right) A^{(i)}_j\right)\right], \tag{13}$$

where $\rho^{(i)}_j = \pi_\theta(a^{(i)}_{(j-1)K:jK-1} \mid h^{(i)}_j)/\pi_{\text{old}}(a^{(i)}_{(j-1)K:jK-1} \mid h^{(i)}_j)$, and $h^{(i)}_j$ denotes the rollout history before the j-th segment in the i-th rollout. By optimizing rewards computed from actual online rollouts, Action-aware GRPO provides direct supervision on action consequences and trains the model to account for how current waypoint decisions influence downstream observations, future actions, and final navigation success. See Appendix A.4 for details.

5 Experiments

We evaluate WorldVLN on outdoor and indoor UAV benchmarks against representative VLN, VLA, and video-based navigation baselines. We analyze training curves and ablate autoregressive modeling and Action-aware GRPO. Finally, we deploy WorldVLN on a real UAV platform to examine its generalization to real-world environments.

Experimental setup: We evaluate WorldVLN on UAV-Flow [41] and IndoorUAV [29], comparing it with the VLA baselines under the corresponding benchmark protocols. WorldVLN uses InfinityStar [27] as the latent autoregressive backbone, with the action decoder initialized from Wan VAE [40] and TSformer-VO-style [13] priors. Training is conducted on 8 NVIDIA A800 80GB GPUs, and simulator rollouts are executed on an RTX 4090 workstation. For the world-model comparison, we reimplement ImagiNav [7] and NavDreamer [18] from their papers, following their reported hyperparameters where applicable. See Appendix A.4 and Appendix A.5 for details.

Table 1: Success rates (SR, %) on UAV-Flow-Sim.

Model	Instruction	Approach	Retreat	Pass	Land	Turn	Move	Shift	Rotate	Surround	A/D	Avg.
Seq2Seq-UAV	Fixed	0.00	0.00	15.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.20
	Open	0.00	0.00	2.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.37
CMA-UAV	Fixed	0.00	91.67	25.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.69
	Open	0.00	91.67	25.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.69
Travel-UAV	Fixed	35.71	100.00	25.00	53.70	20.00	13.33	85.71	46.67	58.33	84.21	52.38
	Open	42.86	100.00	32.50	40.74	6.67	6.67	75.51	40.00	100.00	78.95	50.18
OpenVLA-UAV	Fixed	45.24	100.00	37.50	46.30	73.33	66.67	77.55	20.00	100.00	89.47	59.34
	Open	47.62	100.00	55.00	40.74	73.33	60.00	85.71	6.67	100.00	84.21	61.17
π_0-0-UAV	Fixed	59.52	100.00	50.00	66.67	33.33	60.00	18.37	46.67	58.33	26.32	49.45
	Open	71.43	100.00	62.50	72.22	80.00	80.00	40.82	60.00	75.00	15.79	62.64
ImagiNav	Fixed	83.33	66.67	37.50	74.07	40.00	86.67	28.57	20.00	0.00	42.11	52.01
	Open	80.95	66.67	35.00	74.07	40.00	86.67	28.57	20.00	0.00	42.11	51.28
NavDreamer	Fixed	71.43	91.67	22.50	68.52	6.67	93.33	44.90	6.67	0.00	52.63	49.45
	Open	71.43	91.67	20.00	68.52	6.67	93.33	44.90	6.67	0.00	52.63	49.08
WorldVLN (Ours)	Fixed	97.62	91.67	40.00	92.59	60.00	100.00	85.71	46.67	58.33	94.74	**79.12**
	Open	95.24	91.67	40.00	98.15	53.33	100.00	85.71	33.33	41.67	94.74	**78.02**

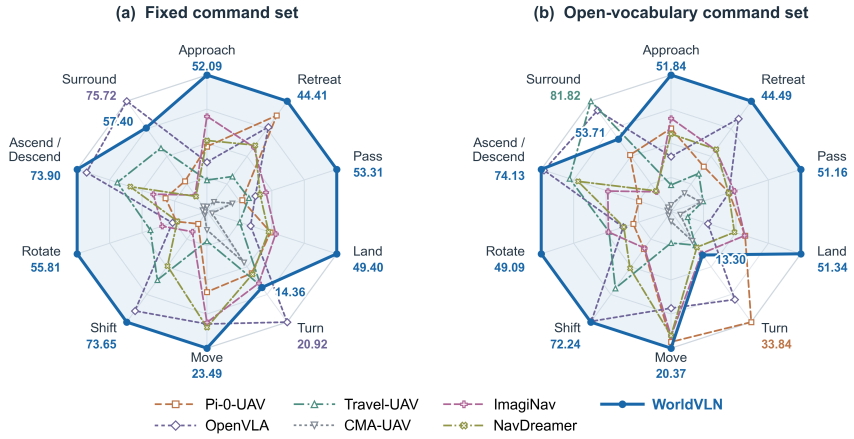


Figure 3: Category-wise NDTW (%) on UAV-Flow-Sim under fixed and open-vocabulary instructions. Spokes use category-specific scales.

5.1 Quantitative results

The quantitative results demonstrate the strong performance of WorldVLN across both outdoor and indoor UAV benchmarks, as listed in Table 1, Figure 3, and Table 2.

- **Performance across benchmarks.** On UAV-Flow-Sim, WorldVLN reaches 79.12% and 78.02% average SR under Fixed and Open instructions, exceeding the strongest baselines by 19.78 and 15.38 percentage points, respectively. Figure 3 compares trajectory agreement with the reference paths using normalized dynamic time warping (NDTW). WorldVLN achieves the highest NDTW among the methods shown in eight of ten categories under both instruction settings. On IndoorUAV-VLA, it achieves 41.76% SR and 13.48% NDTW on the full test set.
- **Comparison with world-model baselines.** With the same InfinityStar-8B backbone, WorldVLN exceeds ImagiNav, the stronger video-based baseline, by 27.11 and 26.74 points in average SR under Fixed and Open instructions on UAV-Flow-Sim. On IndoorUAV-VLA, it leads ImagiNav by 14.29 points in full-set SR and 4.37 points in full-set NDTW. These comparisons evaluate the complete WorldVLN system, and Figure 5 examines autoregressive modeling and Action-aware GRPO separately.
- **Performance by task.** On IndoorUAV-VLA, WorldVLN exceeds the strongest baselines in SR by 7.58 points on Medium and 18.38 points on Hard. On UAV-Flow-Sim, it performs especially well on *Approach*, *Land*, *Move*, *Shift*, and *Ascend/Descend*.

Table 2: Results on the IndoorUAV-VLA benchmark. We report Success Rate (%) and NDTW (%) on three difficulty splits, as well as the **Average** results on the full test set. See Appendix A.7.

Model	Easy		Medium		Hard		Average	
	SR/%↑	NDTW/%↑	SR/%↑	NDTW/%↑	SR/%↑	NDTW/%↑	SR/%↑	NDTW/%↑
GPT-4o	30.30	12.30	4.00	6.57	1.96	4.84	11.69	9.20
Seq2Seq	1.60	2.63	1.20	2.74	1.03	3.03	1.33	2.74
CMA	1.28	1.75	0.75	1.94	1.03	2.01	0.99	1.88
OpenVLA	22.52	2.89	1.19	1.12	0.00	0.12	7.81	2.42
π_0-FAST	18.09	8.83	5.26	2.93	1.14	2.68	8.62	4.71
NaVid	25.31	13.10	18.21	3.21	2.31	1.72	15.82	5.28
π_0	46.58	14.52	21.64	7.64	7.55	4.27	27.16	9.44
ImagiNav	43.30	10.66	30.14	9.41	22.81	8.63	27.47	9.11
NavDreamer	25.33	9.01	26.12	8.65	21.00	8.20	23.04	8.43
WorldVLN (Ours)	49.43	13.04	37.72	14.52	41.19	12.80	**41.76**	**13.48**

Failure analysis. The aggregate gains do not extend to every category. For example, under Open instructions, WorldVLN achieves 40.00%, 33.33%, and 41.67% SR on *Pass*, *Rotate*, and *Surround*, respectively, compared with the best baseline results of 62.50%, 60.00%, and 100.00%. These tasks require precise side selection, yaw changes, or circular radius and direction. One possible explanation is that explicit geometric language constraints must pass through the predicted latent before action decoding, whereas VLA policies can condition action generation directly on language. In addition, observation feedback updates the policy only between action segments, leaving deviations within a segment uncorrected until the next update. These are hypotheses about the current implementation; our autoregressive ablation supports the benefit of feedback but does not isolate these failure mechanisms.

5.2 Case analysis

Figure 4 qualitatively compares WorldVLN with representative VLA baselines in outdoor and indoor scenarios. In the outdoor case, the instruction requires the UAV to interact with the car. OpenVLA-UAV moves directly toward the vehicle and fails to execute a precise spatial maneuver around it. In contrast, WorldVLN correctly grounds the car as the target landmark and generates a smoother trajectory with accurate relative positioning. In the indoor case, the instruction requires the agent to approach the staircase and turn left toward the brown wall. The π_0-IndoorUAV baseline fails to maintain the intended spatial relation with the staircase and wall. WorldVLN consistently identifies the relevant landmarks, approaches the staircase, and performs the left-turn behavior in accordance with the instructed spatial layout. These cases suggest that latent world-action prediction enables more accurate spatial grounding and waypoint generation than direct VLA-style action mapping.

5.3 Ablation study

Does WAM learn more efficiently than VLA? We train OpenVLA from scratch on UAV-Flow and compare its training dynamics with WorldVLN, as shown in Figure 5(a). WorldVLN after Stage-1 supervised training reaches higher success rates under the same step budget than OpenVLA-SFT.

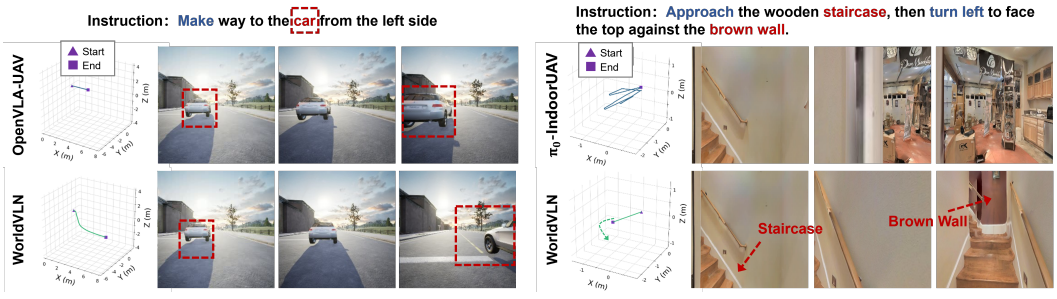


Figure 4: **Qualitative case analysis**. Compared with VLA baselines, WorldVLN shows stronger spatial grounding and more accurate waypoint actions in both outdoor object-centric maneuvers and indoor landmark navigation.

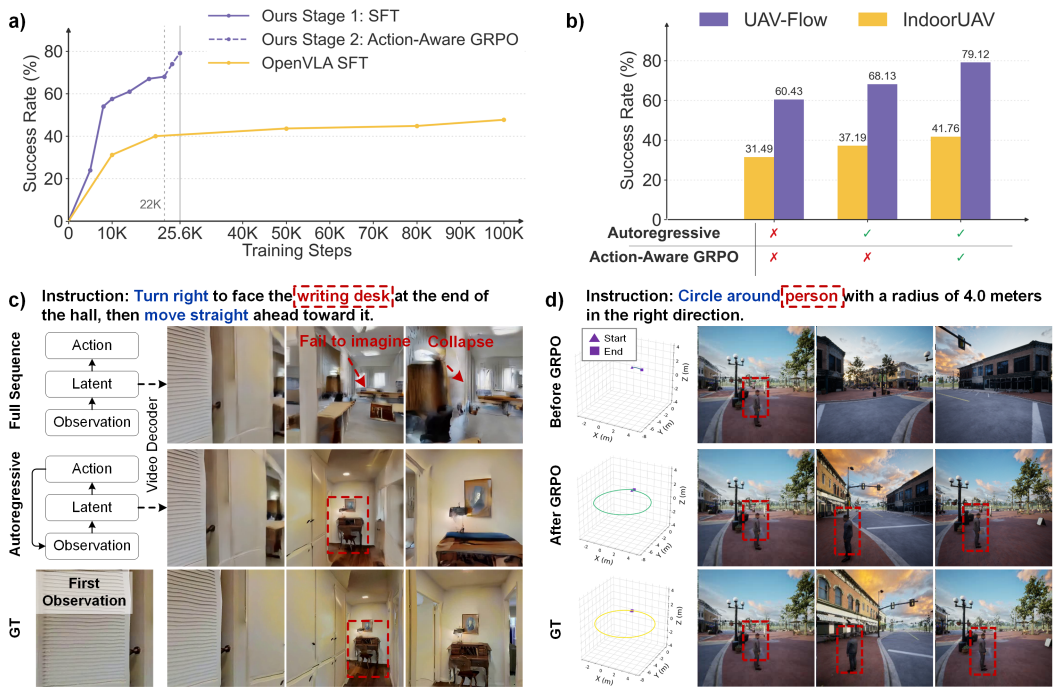


Figure 5: **Ablation studies**. a) Training dynamics compared with OpenVLA on UAV-Flow. b) Quantitative effects of autoregressive modeling and Action-aware GRPO on UAV-Flow and IndoorUAV. c) Latent prediction probe: autoregressive updating preserves more coherent visual-spatial representations than full-sequence prediction. d) Action-aware GRPO improves spatial action accuracy, producing a trajectory closer to the intended circular maneuver.

This suggests that the WAM formulation provides a more effective learning structure for aerial VLN than direct VLA-style observation-to-action mapping.

Why is autoregressive prediction necessary? To isolate the effect of autoregressive modeling, we use the same backbone and action decoder, and compare full-sequence SFT with autoregressive SFT. Figure 5(b) quantitatively shows that autoregressive world-action modeling improves SR by 7.70 points on UAV-Flow and 5.70 points on IndoorUAV. To probe the learned latent representations, Figure 5(c) decodes predicted latents into visual observations for visualization only. The full-sequence variant exhibits semantic drift and scene collapse, indicating unstable long-horizon latent prediction. In contrast, the autoregressive variant repeatedly incorporates newly observed states and preserves more coherent visual-spatial representations, including the instruction-relevant landmark. This suggests that closed-loop autoregressive updating improves latent world prediction and provides more reliable representations for action decoding.

What does Action-aware GRPO learn? To evaluate the effect of action-aware GRPO, we compare the model trained only with Stage-1 supervised training and the model trained with the full two-stage framework. Figure 5(b) shows that adding Action-aware GRPO further boosts performance on both benchmarks. This is also reflected in Figure 5(a), where Action-aware GRPO yields an additional gain of over 10 points after the Stage-1 SFT performance has nearly saturated. Moreover, Figure 5(d) visualizes navigation behavior before and after RL. Before Action-aware GRPO, the model fails to execute a geometrically accurate circular trajectory. After RL, the model produces a trajectory that better follows the intended "circle around" behavior and more closely matches the ground-truth path. This indicates that Action-aware GRPO teaches the model to optimize action consequences beyond visual plausibility, improving action accuracy and goal-directed behavior.

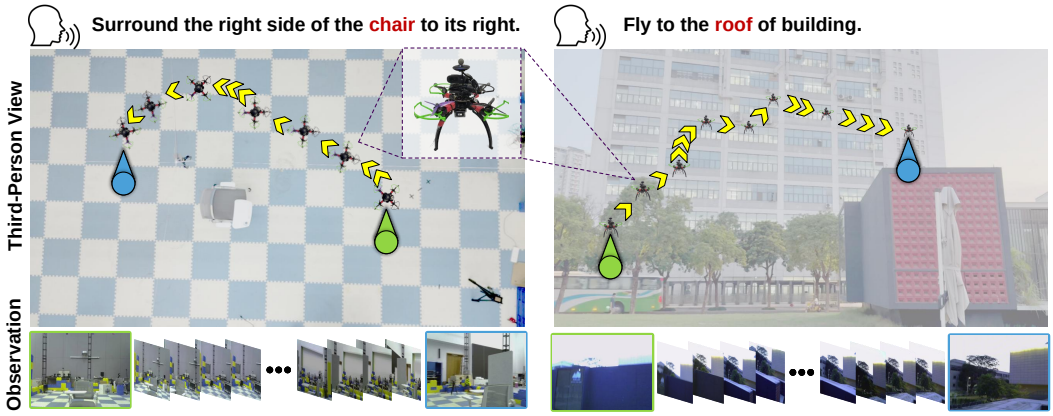


Figure 6: **Real-world UAV deployment**. WorldVLN is trained only in simulation and tested zero-shot on a real drone in both indoor and outdoor scenarios.

5.4 Zero-shot generalization in real-world deployment

To evaluate WorldVLN in real-world environments, we deploy it on a self-built quadrotor with a 250 mm wheelbase, equipped with a Logi C270 RGB camera, a Jetson Orin NX 16GB onboard computer, and a CUAV PX4 flight controller. The WorldVLN policy runs on a remote server: RGB observations are transmitted from the UAV to the server, and predicted waypoint actions are sent back for execution. We conduct indoor tests in a $10\,\mathrm{m} \times 15\,\mathrm{m} \times 3\,\mathrm{m}$ arena with a 14-camera MoCap system, and outdoor tests in an open area using GPS with a TFmini-S LiDAR for altitude estimation.

Figure 6 shows two representative real-world cases. Although WorldVLN is trained only with simulator data, it can follow language instructions and generate executable waypoint actions on the real UAV platform. The indoor case requires the UAV to fly around the right side of a chair in a confined room. This setting is challenging because the agent must rely on close-range visual landmarks and avoid large viewpoint deviations. The outdoor case further investigates the model's ability to navigate in the vertical direction. These results provide evidence that the learned world-action representation can transfer from simulation to real-world UAV deployment even without additional real-world fine-tuning. See Appendix A.9 for details.

6 Conclusions, Limitations, and Future Work

We present WorldVLN, the first autoregressive world action model for aerial vision-language navigation. We introduce a two-stage training framework that first grounds the video prior in instruction-conditioned navigation dynamics and then develops Action-aware GRPO to align the model with action-level navigation outcomes. WorldVLN achieves the highest aggregate success rates on both benchmarks. On IndoorUAV-VLA, its largest SR gain is on the Hard split. On UAV-Flow-Sim, it exceeds both video-based navigation baselines using the same backbone. Real-world UAV deployment further provides promising evidence of zero-shot transfer. Together, WorldVLN provides a concise implicit-prediction architecture and an Action-aware GRPO training strategy, offering a promising route for spatial action tasks and potentially broader embodied domains such as robotic manipulation.

Limitations. Our experiments are validated on relatively short-range aerial navigation, while long-horizon VLN remains to be further explored. Real-world deployment also relies on server-side inference due to the computational cost of the backbone, limiting fully onboard execution.

Future work. We will explore more scalable architectures for long-horizon latent prediction, as well as model compression and inference acceleration for fully onboard UAV deployment.

Acknowledgments and Disclosure of Funding

This work was supported by the Natural Science Foundation of China under Grant 62371269, Shenzhen Low-Altitude Airspace Strategic Program Portfolio (Grant No. Z25306110), Shenzhen Science and Technology Program ZDCYKCX20250901094203005, ZDCY202517012 and Tsinghua University-Toyota Research Center.

References

[1] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3674–3683, 2018.

[2] Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 15791–15801, 2025.

[3] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale, 2023. URL https://arxiv.org/abs/2212.06817.

[4] Yevgen Chebotar, Quan Vuong, Karol Hausman, Fei Xia, Yao Lu, Alex Irpan, Aviral Kumar, Tianhe Yu, Alexander Herzog, Karl Pertsch, et al. Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions. In *Conference on Robot Learning*, pages 3909–3928. PMLR, 2023.

[5] Boyuan Chen, Tianyuan Zhang, Haoran Geng, Kiwhan Song, Caiyi Zhang, Peihao Li, William T. Freeman, Jitendra Malik, Pieter Abbeel, Russ Tedrake, Vincent Sitzmann, and Yilun Du. Large video planner enables generalizable robot control, 2025. URL https://arxiv.org/abs/2512.15840.

[6] Haoxin Chen, Menghan Xia, Yingqing He, Yong Zhang, Xiaodong Cun, Shaoshu Yang, Jinbo Xing, Yaofang Liu, Qifeng Chen, Xintao Wang, Chao Weng, and Ying Shan. Videocrafter1: Open diffusion models for high-quality video generation, 2023. URL https://arxiv.org/abs/2310.19512.

[7] Jie Chen, Yuxin Cai, Yizhuo Wang, Ruofei Bai, Yuhong Cao, Jun Li, Wei-Yun Yau, and Guillaume Sartoretti. Imaginav: Scalable embodied navigation via generative visual prediction and inverse dynamics, 2026. URL https://arxiv.org/abs/2603.13833.

[8] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[9] Jordan Crivelli-Decker, Alex Clarke, Seongmin A Park, Derek J Huffman, Erie D Boorman, and Charan Ranganath. Goal-oriented representations in the human hippocampus during planning and navigation. *Nature communications*, 14(1):2946, 2023.

[10] Yousef Emami, Hao Zhou, Luis Almeida, and Kai Li. Diffusion models for smarter uavs: Decision-making and modeling, 2025. URL https://arxiv.org/abs/2501.05819.

[11] Russell A Epstein, Eva Zita Patai, Joshua B Julian, and Hugo J Spiers. The cognitive map in humans: spatial navigation and beyond. *Nature neuroscience*, 20(11):1504–1513, 2017.

[12] Uğur M Erdem and Michael Hasselmo. A goal-directed spatial navigation model using forward trajectory planning based on grid cells. *European Journal of Neuroscience*, 35(6):916–931, 2012.

[13] André O Françani and Marcos ROA Maximo. Transformer-based model for monocular visual odometry: a video understanding approach. *IEEE Access*, 13:13959–13971, 2025.

[14] Yunpeng Gao, Chenhui Li, Zhongrui You, Junli Liu, Zhen Li, Pengan Chen, Qizhi Chen, Zhonghan Tang, Liansheng Wang, Penghui Yang, Yiwen Tang, Yuhang Tang, Shuai Liang, Songyi Zhu, Ziqin Xiong, Yifei Su, Xinyi Ye, Jianan Li, Yan Ding, Dong Wang, Xuelong Li, Zhigang Wang, and Bin Zhao. Openfly: A comprehensive platform for aerial vision-language navigation, 2026. URL https://arxiv.org/abs/2502.18041.

[15] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[16] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. Imagen video: High definition video generation with diffusion models, 2022. URL https://arxiv.org/abs/2210.02303.

[17] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David Fleet. Video diffusion models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 8633–8646. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/39235c56aef13fb05a6adc95eb9d8d66-Paper-Conference.pdf.

[18] Xijie Huang, Weiqi Gai, Tianyue Wu, Congyu Wang, Qiaoyu Zheng, Zhiyang Liu, Xin Zhou, Yuze Wu, and Fei Gao. Navdreamer: Video models as zero-shot 3d navigators. *IEEE Robotics and Automation Letters*, 11(9):10513–10520, 2026. doi: 10.1109/LRA.2026.3713727.

[19] Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, Vedant Choudhary, Foster Collins, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Maitrayee Dhaka, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachlan Groom, Haroun Habeeb, Hunter Hancock, Karol Hausman, Gashon Hussein, Victor Hwang, Brian Ichter, Connor Jacobsen, Szymon Jakubczak, Rowan Jen, Tim Jones, Gregg Kammerer, Ben Katz, Liyiming Ke, Mairbek Khadikov, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Brendon LeCount, Sergey Levine, Xinyu Li, Adrian Li-Bell, Vladislav Lialin, Zhonglin Liang, Wallace Lim, Yao Lu, Enyu Luo, Vishnu Mano, Nandan Marwaha, Aikys Mongush, Liam Murphy, Suraj Nair, Tyler Patterson, Karl Pertsch, Allen Z. Ren, Gavin Schelske, Charvi Sharma, Baifeng Shi, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Jiaming Tang, Jimmy Tanner, Shalom Tekeste, Marcel Torne, Kyle Vedder, Quan Vuong, Anna Walling, Haohuan Wang, Jason Wang, XuDong Wang, Chris Whalen, Samuel Whitmore, Blake Williams, Charles Xu, Sukwon Yoo, Lili Yu, Wuming Zhang, Zhuoyang Zhang, and Ury Zhilinsky. $\pi_{0.7}$: a steerable generalist robotic foundation model with emergent capabilities, 2026. URL https://arxiv.org/abs/2604.15483.

[20] Bingyi Kang, Yang Yue, Rui Lu, Zhijie Lin, Yang Zhao, Kaixin Wang, Gao Huang, and Jiashi Feng. How far is video generation from world model: A physical law perspective, 2025. URL https://arxiv.org/abs/2411.02385.

[21] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model, 2024. URL https://arxiv.org/abs/2406.09246.

[22] Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video

models for visuomotor control and planning, 2026. URL https://arxiv.org/abs/2601.16163.

[23] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *European Conference on Computer Vision*, pages 104–120. Springer, 2020.

[24] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33: 1179–1191, 2020.

[25] Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal world modeling for robot control, 2026. URL https://arxiv.org/abs/2601.21998.

[26] Kunyang Lin, Peihao Chen, Diwei Huang, Thomas H. Li, Mingkui Tan, and Chuang Gan. Learning vision-and-language navigation from youtube videos. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8283–8292, 2023. doi: 10.1109/ICCV51070.2023.00764.

[27] Jinlai Liu, Jian Han, Bin Yan, Hui Wu, Fengda Zhu, Xing Wang, Yi Jiang, Bingyue Peng, and Zehuan Yuan. Infinitystar: Unified spacetime autoregressive modeling for visual generation. *arXiv preprint arXiv:2511.04675*, 2025.

[28] Shubo Liu, Hongsheng Zhang, Yuankai Qi, Peng Wang, Yanning Zhang, and Qi Wu. Aerialvln: Vision-and-language navigation for uavs. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15338–15348, 2023. doi: 10.1109/ICCV51070.2023.01411.

[29] Xu Liu, Yu Liu, Hanshuo Qiu, Yang Qirong, and Zhouhui Lian. Indooruav: Benchmarking vision-language uav navigation in continuous indoor environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 23864–23872, 2026.

[30] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.

[31] Jonas Pai, Liam Achenbach, Victoriano Montesinos, Benedek Forrai, Oier Mees, and Elvis Nava. mimic-video: Video-action models for generalizable robot control beyond vlas, 2025. URL https://arxiv.org/abs/2512.15692.

[32] Zhangyang Qi, Zhixiong Zhang, Yizhou Yu, Jiaqi Wang, and Hengshuang Zhao. Vln-r1: Vision-language navigation via reinforcement fine-tuning, 2025. URL https://arxiv.org/abs/2506.17221.

[33] Yiran Qin, Zhelun Shi, Jiwen Yu, Xijun Wang, Enshen Zhou, Lijun Li, Zhenfei Yin, Xihui Liu, Lu Sheng, Jing Shao, Lei Bai, Wanli Ouyang, and Ruimao Zhang. Worldsimbench: Towards video generation models as world simulators, 2024. URL https://arxiv.org/abs/2410.18072.

[34] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[35] Dhruv Shah, Błażej Osiński, Sergey Levine, et al. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. In *Conference on robot learning*, pages 492–504. pmlr, 2023.

[36] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[37] Xinshuai Song, Weixing Chen, Yang Liu, Weikai Chen, Guanbin Li, and Liang Lin. Towards long-horizon vision-language navigation: Platform, benchmark and method. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12078–12088, 2025.

[38] Rich Sutton. The bitter lesson. https://www.cs.utexas.edu/~eunsol/courses/data/bitter_lesson.pdf, 2019. Accessed: 2026-05-07.

[39] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models, 2025. URL https://arxiv.org/abs/2503.20314.

[40] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[41] Xiangyu Wang, Donglin Yang, Yue Liao, Wenhao Zheng, Bin Dai, Hongsheng Li, Si Liu, et al. Uav-flow colosseo: A real-world benchmark for flying-on-a-word uav imitation learning. *arXiv preprint arXiv:2505.15725*, 2025.

[42] Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, et al. Vla-adapter: An effective paradigm for tiny-scale vision-language-action model. In *Proceedings of the AAAI conference on artificial intelligence*, volume 40, pages 18638–18646, 2026.

[43] Zun Wang, Jialu Li, Yicong Hong, Songze Li, Kunchang Li, Shoubin Yu, Yi Wang, Yu Qiao, Yali Wang, Mohit Bansal, and Limin Wang. Bootstrapping language-guided navigation learning with self-refining data flywheel, 2025. URL https://arxiv.org/abs/2412.08467.

[44] Bing Wu, Chang Zou, Changlin Li, Duojun Huang, Fang Yang, Hao Tan, Jack Peng, Jianbing Wu, Jiangfeng Xiong, Jie Jiang, Linus, Patrol, Peizhen Zhang, Peng Chen, Penghao Zhao, Qi Tian, Songtao Liu, Weijie Kong, Weiyan Wang, Xiao He, Xin Li, Xinchi Deng, Xuefei Zhe, Yang Li, Yanxin Long, Yuanbo Peng, Yue Wu, Yuhong Liu, Zhenyu Wang, Zuozhuo Dai, Bo Peng, Coopers Li, Gu Gong, Guojian Xiao, Jiahe Tian, Jiaxin Lin, Jie Liu, Jihong Zhang, Jiesong Lian, Kaihang Pan, Lei Wang, Lin Niu, Mingtao Chen, Mingyang Chen, Mingzhe Zheng, Miles Yang, Qiangqiang Hu, Qi Yang, Qiuyong Xiao, Runzhou Wu, Ryan Xu, Rui Yuan, Shanshan Sang, Shisheng Huang, Siruis Gong, Shuo Huang, Weiting Guo, Xiang Yuan, Xiaojia Chen, Xiawei Hu, Wenzhi Sun, Xiele Wu, Xianshun Ren, Xiaoyan Yuan, Xiaoyue Mi, Yepeng Zhang, Yifu Sun, Yiting Lu, Yitong Li, You Huang, Yu Tang, Yixuan Li, Yuhang Deng, Yuan Zhou, Zhichao Hu, Zhiguang Liu, Zhihe Yang, Zilin Yang, Zhenzhi Lu, Zixiang Zhou, and Zhao Zhong. Hunyuanvideo 1.5 technical report, 2025. URL https://arxiv.org/abs/2511.18870.

[45] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In *Conference on robot learning*, pages 2226–2240. PMLR, 2023.

[46] Wansen Wu, Tao Chang, Xinmeng Li, Quanjun Yin, and Yue Hu. Vision-language navigation: a survey and taxonomy. *Neural Computing and Applications*, 36(7):3291–3316, 2024.

[47] Xinda Xue, Junjun Hu, Minghua Luo, Shichao Xie, Jintao Chen, Zixun Xie, Kuichen Quan, Wei Guo, Mu Xu, and Zedong Chu. Omninav: A unified framework for prospective exploration and visual-language navigation, 2026. URL https://arxiv.org/abs/2509.25687.

[48] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Yuxuan Zhang, Weihan Wang, Yean Cheng, Bin Xu, Xiaotao Gu, Yuxiao Dong, and Jie Tang. Cogvideox: Text-to-video diffusion models with an expert transformer, 2025. URL https://arxiv.org/abs/2408.06072.

[49] Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies, 2026. URL https://arxiv.org/abs/2602.15922.

[50] Jiazhao Zhang, Kunyu Wang, Rongtao Xu, Gengze Zhou, Yicong Hong, Xiaomeng Fang, Qi Wu, Zhizheng Zhang, and He Wang. Navid: Video-based vlm plans the next step for vision-and-language navigation. *arXiv preprint arXiv:2402.15852*, 2024.

[51] Kaiwen Zhang, Zhenyu Tang, Xiaotao Hu, Xingang Pan, Xiaoyang Guo, Yuan Liu, Jingwei Huang, Li Yuan, Qian Zhang, Xiao-Xiao Long, et al. Epona: Autoregressive diffusion world model for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 27220–27230, 2025.

[52] Weichen Zhang, Chen Gao, Shiquan Yu, Ruiying Peng, Baining Zhao, Qian Zhang, Jinqiang Cui, Xinlei Chen, and Yong Li. Citynavagent: Aerial vision-and-language navigation with hierarchical semantic planning and global memory. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 31292–31309, 2025.

[53] 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*, 2025.

[54] Baining Zhao, Jianjie Fang, Zichao Dai, Ziyou Wang, Jirong Zha, Weichen Zhang, Chen Gao, Yue Wang, Jinqiang Cui, Xinlei Chen, and Yong Li. UrbanVideo-bench: Benchmarking vision-language models on embodied intelligence with video data in urban spaces. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32400–32423, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1558. URL https://aclanthology.org/2025.acl-long.1558/.

[55] Baining Zhao, Ziyou Wang, Jianjie Fang, Chen Gao, Fanhang Man, Jinqiang Cui, Xin Wang, Xinlei Chen, Yong Li, and Wenwu Zhu. Embodied-r: Collaborative framework for activating embodied spatial reasoning in foundation models via reinforcement learning. In *Proceedings of the 33rd ACM International Conference on Multimedia*, MM '25, page 11071–11080, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400720352. doi: 10.1145/3746027.3755703. URL https://doi.org/10.1145/3746027.3755703.

[56] Gengze Zhou, Yicong Hong, and Qi Wu. Navgpt: Explicit reasoning in vision-and-language navigation with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 7641–7649, 2024.

A Appendix

A.1 Broader Impacts and Responsible Deployment

WorldVLN may benefit UAV-based embodied navigation applications such as infrastructure inspection, search and rescue, and disaster assessment. In these scenarios, language-conditioned aerial agents can reduce the cost of manual operation and decrease the need for humans to enter hazardous or hard-to-access environments. However, autonomous UAV navigation may also introduce potential risks, including physical safety concerns, privacy violations, unauthorized surveillance, and misuse in restricted or safety-sensitive areas. Therefore, the results in this paper should not be interpreted as evidence that the system is ready for unsupervised deployment in public or safety-critical environments.

Our real-world experiments are conducted only in controlled indoor arenas or relatively enclosed outdoor areas. Low-level flight stabilization is handled by the PX4 flight controller, while high-level model inference and monitoring are performed through a ground-station server. Practical deployment should include human supervision, geofencing, speed and altitude limits, emergency stop mechanisms, and reliable state-estimation checks, and should comply with local UAV regulations. For code and model release, we recommend that they be used primarily for research and simulator evaluation; real-world UAV deployment interfaces should only be used after sufficient safety validation and under appropriate hardware supervision.

A.2 Limitations

Although WorldVLN achieves strong results on both indoor and outdoor UAV benchmarks, the current study mainly focuses on short-range aerial navigation and short-horizon waypoint generation. Therefore, the capability of the model remains to be further validated in long-horizon VLN scenarios, multi-stage complex instructions, large-scale outdoor exploration, and long-term closed-loop decision-making.

In addition, WorldVLN is mainly trained on simulator data and public benchmark trajectories, and the real-world evaluation is conducted only in controlled indoor arenas and relatively enclosed outdoor areas. Its robustness under more challenging real-world conditions, such as strong illumination changes, adverse weather, dynamic obstacles, crowded spaces, or GPS-denied environments, has not been fully evaluated. Due to the large scale of the autoregressive world backbone, the current real-world deployment still relies on server-side inference and cannot yet run fully onboard. Moreover, we do not conduct extensive multi-seed experiments in this work. Future work should further improve model compression, inference acceleration, safety constraints, and statistical validation.

A.3 Details of model architecture

WorldVLN adopts a latent-space spatiotemporal autoregressive architecture as its world-model backbone. Following the InfinityStar [27] and the WAN VAE [40], the overall architecture consists of a text encoder, a discrete video tokenizer, a spatiotemporal autoregressive Transformer, and an action decoder. As shown in Figure 7, given a language instruction and egocentric visual observations, the text encoder converts the instruction into text tokens, while the visual observations are converted by the video tokenizer into known visual pyramid conditions. The spatiotemporal autoregressive Transformer then predicts the token blocks of future target clips conditioned on both the text tokens and the known visual token conditions. These predicted token blocks are further aggregated into future world-state latent representations, which are used as the input to the action decoder for action generation.

The visual tokenizer uses the video VAE encoder from the adopted pretrained tokenizer as the continuous visual compression module. Specifically, the input image or video is first encoded into a compact latent representation. Multi-scale residual quantization is then performed on this latent representation to obtain a set of discrete residual token blocks. For a single-frame image condition, these token blocks are organized as an image pyramid; for multi-frame historical video conditions, they are organized as historical clip pyramids. This representation unifies visual observations into multi-scale token conditions, enabling both image inputs and historical video inputs to be processed by the same spatiotemporal autoregressive Transformer.

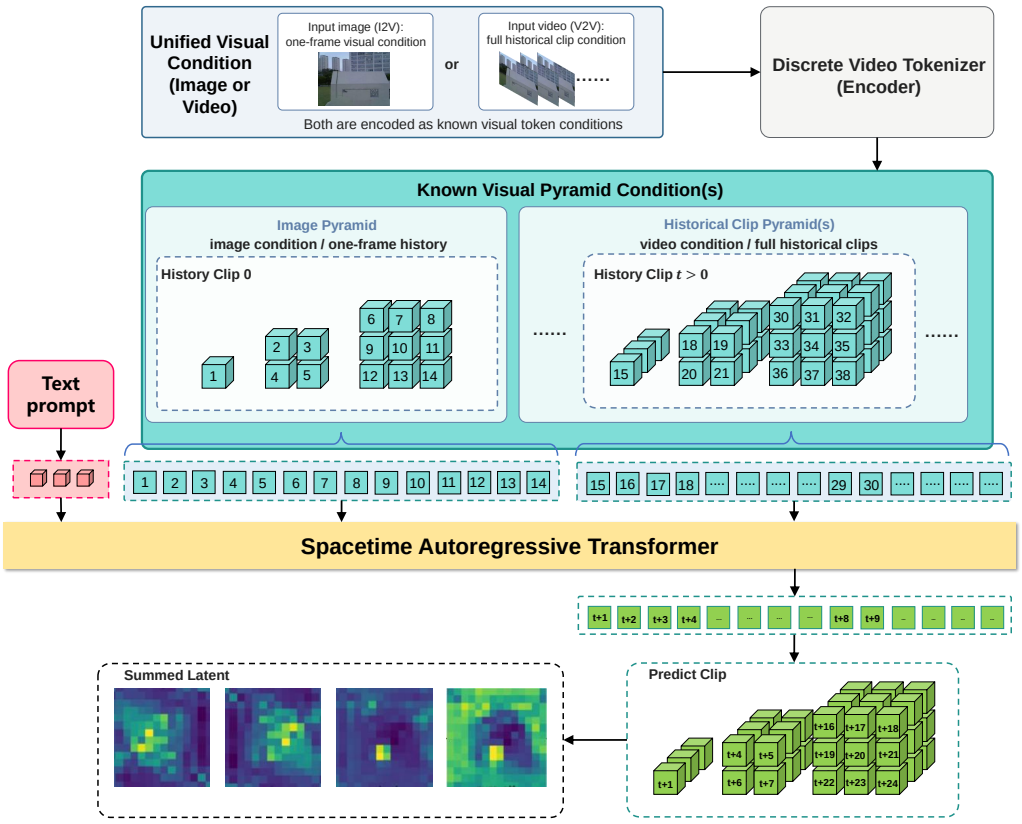


Figure 7: Architecture of the latent-space spatiotemporal autoregressive world backbone. The input image or historical video is encoded into known visual pyramid conditions, which are used together with text tokens to predict future target clip pyramids. The predicted token blocks are aggregated into output latent representations for subsequent action decoding.

The spatiotemporal autoregressive Transformer predicts future target clip pyramids from the known visual pyramid conditions. Within each target clip, the model follows a coarse-to-fine scale order for next-scale prediction: it first predicts low-resolution token blocks that mainly capture global structure, and then progressively predicts higher-resolution token blocks that provide local details. Along the temporal dimension, the model performs clip-order autoregression: the first target clip is predicted from the known visual conditions, and subsequent target clips are predicted conditioned on preceding target clips. Therefore, this architecture jointly models the spatial scale dependency within each clip and the temporal evolution across clips. After prediction, the multi-scale token blocks of each target clip are merged into the corresponding latent representation. This latent representation is not decoded as the final RGB video output; instead, it is treated as a future world-state representation and fed into the action decoder to generate low-level navigation actions.

The action decoder receives the future latent representation produced by the world backbone and converts it into executable low-level navigation actions. As shown in Figure 8, the module takes the world-model output latent as input and regards it as a compact spatiotemporal representation that contains short-horizon future state changes. This latent representation encodes spatiotemporal information related to viewpoint changes, spatial-structure changes, and motion trends within the prediction horizon, and thus provides a direct basis for action inference. The action decoder finally outputs a continuous action vector, where each action corresponds to the relative 3D displacement and relative yaw change of the UAV.

Structurally, the action decoder consists of a vision embedding module, a spatiotemporal Transformer backbone, and an action regression head. The vision embedding module first reorganizes the input segment-level latent through feature reshaping, convolutional mapping, upsampling, and projection, converting it into unified spatiotemporal embedding tokens while preserving both the temporal order

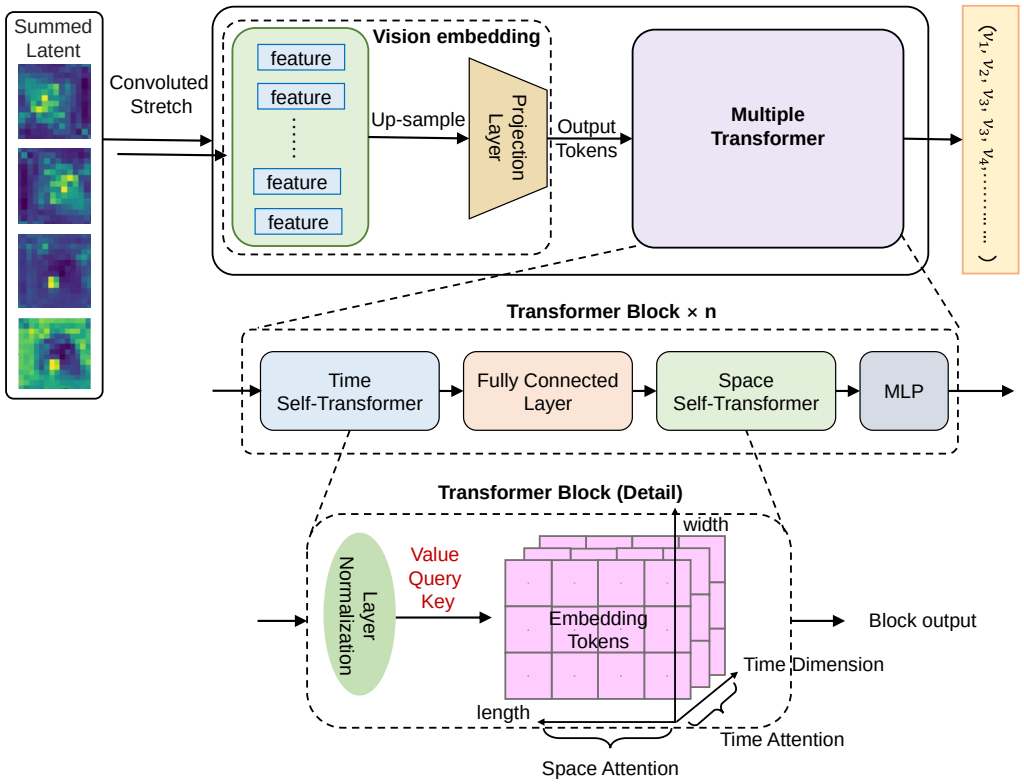


Figure 8: Architecture of the action decoder. The world-model output latent is first converted into spatiotemporal embedding tokens by the vision embedding module. Multiple Transformer blocks with factorized temporal and spatial attention then model action-related spatiotemporal features, which are finally regressed into continuous UAV navigation actions.

and spatial layout of the latent representation. Then, multiple Transformer blocks perform action-related feature modeling with factorized spatiotemporal attention: temporal attention captures motion evolution and viewpoint changes across latent frames, while spatial attention models the geometric structure and spatial relations within each latent frame. Finally, the aggregated spatiotemporal representation is regressed by an MLP action head into continuous action vectors, enabling the action decoder to infer the UAV navigation actions directly from the future latent state changes predicted by the world model.

A.4 Details of training framework

Supervised fine-tuning of the World Backbone. WorldVLN adopts a spacetime autoregressive video prediction setup during supervised fine-tuning. Given a navigation instruction and its corresponding egocentric navigation video, we divide the video into one initial segment and multiple future segments. The initial segment contains only the frame 0 and provides the initial observation condition, while each future segment contains $K = 16$ frames and represents a short-horizon future observation segment. The temporal compression ratio of the video tokenizer is set to 4; therefore, each 16-frame clip corresponds to 4 temporal steps in the latent/token space. Each training sample contains 49 frames in total, consisting of one initial frame and three consecutive 16-frame future segments. This configuration aligns the temporal granularity of the world model with the short-horizon decision window required for navigation.

When training the autoregressive world backbone, we adopt segment-level supervised fine-tuning. For each target future segment, the model is conditioned on the language instruction and the complete ground-truth observation history preceding that segment, and predicts the discrete multi-scale token representation of the current clip. The model does not use its own previously generated clips as historical context during this stage. We retain the video decoder as a training-time visual supervision

interface, so that the backbone is optimized toward future visual predictions that remain decodable by the video decoder. Through this first-stage SFT, the model learns to predict plausible short-horizon future visual changes conditioned on the instruction and real historical observations, thereby forming a latent imagination capability for navigation. During inference, the video decoder is not used for action generation; instead, the predicted latent world transition is directly fed into the subsequent action decoder.

Supervised training of the Action Decoder. The video-to-action teacher follows the TSformer-VO-style visual odometry/action decoding backbone, while the latent projection is initialized with priors from the adopted video decoder. The training of the Action Decoder starts from a video-to-action teacher model. We first train this teacher model using real video clips paired with expert action sequences, enabling it to learn the mapping from continuous visual observations to navigation actions. The teacher model then provides action-aware spatiotemporal representation supervision for the latent-space Action Decoder, which alleviates the representation alignment difficulty caused by directly learning action prediction from compressed latents.

Based on this teacher model, we transfer the action decoding process into the latent space of the world model. Specifically, each real video clip is first encoded by the frozen video encoder to obtain the ground-truth video latent. We then use the embedding/token representation produced by the pretrained video-to-action teacher model as the distillation target, and align the Vision Embedding module of the Action Decoder to perform latent-to-token representation mapping, so that its output can be compatible with the subsequent action decoding backbone. After this representation alignment, we train the entire Action Decoder with ground-truth video latents and their corresponding expert action sequences, ultimately establishing the mapping from latent world representations to continuous navigation actions.

Action-aware GRPO. We provide the reward implementation details used in Action-aware GRPO. The formulation follows the segment-wise notation in the main text and is applied to both UAV-Flow and IndoorUAV-VLA, with benchmark-specific action formats and success thresholds. For each navigation case, the current policy samples a group of $G = 4$ online autoregressive rollouts. The i-th rollout contains multiple decision segments, and the j-th action segment is denoted by $a^{(i)}_{(j-1)K:jK-1}$. Each rollout starts from the given initial pose and observation. At each segment, the model predicts a short-horizon latent world transition, decodes it into waypoint actions, executes the actions in the environment, receives the next observation segment, and updates the autoregressive context. Future frames are not provided during rollout.

We use three reward terms: an trajectory-consistency reward, a task-progress reward, and a CE-style reference alignment reward. In our experiments, we set the weights of the three reward terms as:

$$\lambda_{\text{traj}} = 0.3, \qquad \lambda_{\text{task}} = 1.3, \qquad \lambda_{\text{ref}} = 0.03. \tag{14}$$

Trajectory reward. For each segment, we first compute the action error between the sampled action segment and the expert action segment. The trajectory distance is defined as

$$d^{(i)}_{\text{traj},j} = 0.45\,\text{MSE}^{(i)}_{xyz,j} + 0.45\,\text{MSE}^{(i)}_{\text{yaw},j} + 0.1\,\text{MSE}^{(i)}_{\text{all},j}, \tag{15}$$

where MSE_{xyz} measures the translation error, MSE_{yaw} measures the yaw error, and MSE_{all} is computed over the full normalized action representation. Equivalently, $d^{(i)}_{\text{traj},j}$ serves as the implementation of the trajectory distance used in Eq. 10. This term encourages the sampled waypoint segment to remain geometrically consistent with the expert trajectory. We assign a relatively large weight to yaw error because yaw changes directly determine the agent's egocentric field of view and strongly affect subsequent observations, while yaw control is empirically harder to learn than translational displacement.

Task reward. The task reward measures whether the rollout successfully completes the intended navigation goal. We determine task progress mainly by comparing the distance between the rollout endpoint and the target endpoint, starting from the same initial pose. This endpoint-based criterion is applicable to both UAV-Flow and IndoorUAV-VLA, while the specific success thresholds follow the corresponding benchmark protocol. Specifically, we can use both quantitative and qualitative signals to measure the task reward. The quantitative signal is a dense geometric score based on the

Euclidean endpoint distance. The qualitative signal is a binary success indicator determined by the benchmark-specific success rule.

Reference reward. To regularize the updated policy toward the reference behavior, we use a CE-style alignment cost computed from the reference-policy log-probability. We introduce this term because optimizing only the trajectory and task rewards can overly shift the autoregressive backbone away from its original video-generation prior. Empirically, when the fine-tuned backbone is connected back to the video decoder for visualization, the generated frames may show degraded visual consistency or even collapse, indicating that the latent imagination capability has been weakened. The CE-style reference alignment reward mitigates this issue by constraining the updated policy to stay close to the reference behavior, thereby preserving the world-model prior learned during supervised training. Smaller CE costs indicate stronger agreement with the reference policy.

Temporal weighting. We use $\gamma = 0.9$. At segment j, the reward-level factor γ^{j-1} is common to the rollouts and largely cancels under group normalization. We also multiply the normalized advantage by γ^{j-1} so that the clipped GRPO objective gives more weight to earlier segments.

Together, these design choices provide dense credit assignment for the VLN tasks. The main optimization settings are a maximum of 3000 training iterations, learning rate 8×10^{-7}, KL coefficient 0.9, PPO ratio clipping threshold 0.02, gradient clipping 0.5, video batch size 1, and maximum token length 20480. To make GRPO fine-tuning memory-efficient for the 8B world backbone, we use partial freezing with the first five model chunks frozen.

A.5 Details of experimental setup

To systematically evaluate the effectiveness of WorldVLN on UAV vision-language action generation, we conduct experiments on two complementary benchmarks: UAV-Flow and IndoorUAV-VLA. UAV-Flow follows the Flying-on-a-Word task formulation, where the model generates low-level flight actions conditioned on egocentric observations, UAV states, and atomic language instructions. It further evaluates the model's ability to capture flight intent, visual spatial relations, and dynamically feasible flight trajectories under language-conditioned UAV control. IndoorUAV-VLA is the indoor VLA subset of IndoorUAV, constructed by segmenting long-horizon indoor navigation trajectories into short sub-trajectories. Each instruction typically corresponds to 1–3 local UAV actions, which enables evaluation of local spatial understanding, orientation control, and fine-grained action generation in continuous 3D indoor environments. Together, these two benchmarks provide complementary evaluation settings for vision-language navigation.

UAV-Flow-Sim covers ten flight skill categories under fixed-template and open-vocabulary instructions (Figure 9). IndoorUAV-VLA groups episodes into Easy, Medium, and Hard splits by the number of action types in the instruction (Figure 10).

Evaluation metrics. The definitions and computation of SR and NDTW follow the respective benchmark protocols in UAV-Flow [41] and IndoorUAV [29].

For both benchmarks, we compare WorldVLN with representative baselines covering different technical paradigms, including traditional VLN methods, UAV-specific or waypoint-based policies, and general VLA models. All methods are evaluated under the corresponding input-output format and evaluation protocol of each benchmark, ensuring fair comparison and enabling a comprehensive analysis of WorldVLN's world-action modeling capability.

World model-based baselines. ImagiNav recovers trajectories from generated videos through inverse dynamics. NavDreamer scores candidate videos with a VLM and recovers waypoints from the selected video. Both reimplementations use the same InfinityStar-8B video backbone as WorldVLN and are evaluated on UAV-Flow-Sim and IndoorUAV-VLA.

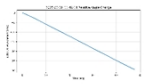
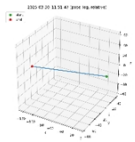
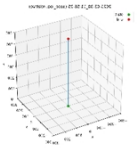
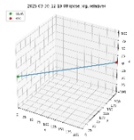
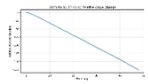
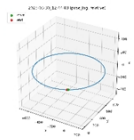
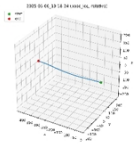
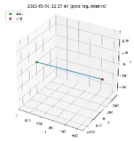
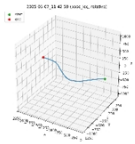
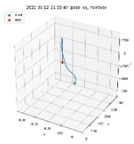
Type	Image1	Image2	Image3	Prompt	Trajectory
Turn				Turn to the direction of the person.	
Move				Navigate to a point 4.5 meters away from the individual	
Ascend				Ascend to an altitude of 6.0 meters	
shift				Move 3.0 meters to the left at a 35-degree angle	
Rotation				Turn counterclockwise by 180 degrees	
Surround				Orbit the person to the right at a 5.0-meter radius.	
Approach				Suggest traveling to the streetlight from the front side	
Retreat				Move backward from the sculpture ahead	
Pass				Request to move through the streetlight from the right side	
Land				Please land at the sculpture on the right side	

Figure 9: Qualitative examples from the UAV-Flow benchmark. The benchmark covers diverse fine-grained UAV flight actions, including target-oriented motion, primitive translation, vertical control, and object-relative navigation.

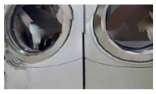
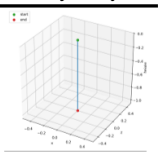
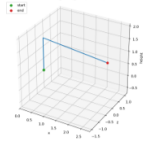
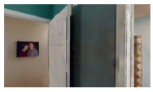
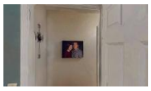
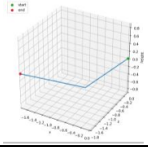
Type	Image1	Image2	Image3	Prompt	Trajectory
Easy				Descend toward the washing machine.	
Medium				Rise up to frame the pair of chairs by the window, then move forward through the doorway.	
Hard				Fly forward toward the toilet, then turn left to the photograph on the wall and fly forward to it.	

Figure 10: Qualitative examples from the IndoorUAV-VLA benchmark. Easy, Medium, and Hard correspond to increasing action-composition complexity, where the UAV needs to execute one, two, or three types of low-level actions.

Existing assets and licenses. Our experiments use public benchmarks, pretrained models, and open-source software under their original protocols and licenses. Specifically, UAV-Flow and IndoorUAV-VLA are used for training and evaluation; InfinityStar-8B and the video VAE serve as pretrained backbone/tokenizer components; and PX4, MAVLink, and VRPN support real-world flight control, communication, and external pose integration. We cite the corresponding papers or project pages and follow their terms of use.

A.6 Details of results on UAV-Flow

UAV-Flow evaluates fine-grained language-conditioned UAV control under both fixed-template and open-vocabulary instruction settings. Our model achieves 79.12% average SR under the Fixed setting and 78.02% average SR under the Open setting, with only a small performance gap between the two settings, demonstrating that the model can robustly map different language expressions to consistent low-level flight behaviors. Specifically, we improve the performance on **Approach** to 97.62% Fixed SR and 95.24% Open SR, showing that the model can effectively couple target localization, distance estimation, and target-proximity control. We improve the performance on **Land** to 92.59% Fixed SR and 98.15% Open SR, indicating that the model can accurately control both horizontal target alignment and vertical descent, which verifies its strong 3D terminal-state control ability. We further achieve 100.00% SR on **Move**, 85.71% SR on **Shift**, and 94.74% SR on **A/D** under both instruction settings, demonstrating reliable primitive translation, egocentric lateral control, and vertical degree-of-freedom control. These results show that our model is particularly effective in aligning language instructions, visual targets, and low-level UAV motion, especially for target proximity, precise landing, basic translation, sideward movement, and height control.

A.7 Details of results on IndoorUAV

IndoorUAV-VLA evaluates short-horizon indoor UAV control using SR and NDTW, where SR measures whether the UAV successfully reaches the target pose and NDTW measures trajectory-level consistency with the reference path. Our model achieves 41.76% SR and 13.48% NDTW on the Full split, showing that it improves both final-pose success and trajectory quality. More importantly, we achieve 37.72% SR and 14.52% NDTW on the Medium split, indicating that the model can better compose multiple low-level actions within a short trajectory rather than only executing a single primitive motion. On the Hard split, our model further achieves 41.19% SR and 12.80% NDTW, demonstrating stronger multi-step coordination when translation, rotation, vertical movement, and other motion primitives must be jointly executed. These improvements show that the main advantage of our method lies in compositional short-horizon UAV control, where online autoregressive GRPO

rollouts and closed-loop world-action updating help the model maintain state consistency and correct intermediate decisions across multiple action steps.

A.8 Details of VLA vs WAM

To clarify the experimental setup for comparing VLA and WAM, we use OpenVLA as a representative VLA baseline and WorldVLN as an autoregressive world action model. Both models are trained and evaluated on exactly the same training and test splits of UAV-Flow-Sim, and use the same waypoint/action normalization. The SR evaluation protocol also follows the original benchmark setting. This setup controls for factors such as data splits, action normalization, and evaluation criteria, allowing the comparison to focus more directly on the difference between the two modeling paradigms: OpenVLA follows a direct observation-to-action formulation, whereas WorldVLN first performs latent world prediction and then generates actions through an action decoder.

In terms of configuration, we follow the setting in the UAV-Flow benchmark for the OpenVLA training. Both models are trained with the AdamW optimizer and bf16 precision. The learning rate of WorldVLN is set to 1×10^{-5}, with a maximum training budget of 26K steps. The learning rate of OpenVLA is set to 5×10^{-4}, with a maximum training budget of 100K steps. For batch size, both models are trained on 8 GPUs. OpenVLA uses a per-GPU batch size of 1, resulting in a fixed global batch size of 8. WorldVLN uses token-budget-based sequence packing; under the current 49-frame configuration, its effective global batch size is approximately 8 clips per step. Overall, this setup preserves the native training configuration of each model family under the same data and evaluation protocol, enabling a comparison between VLA and WAM on UAV-Flow-Sim.

A.9 Details of Real-World Deployment

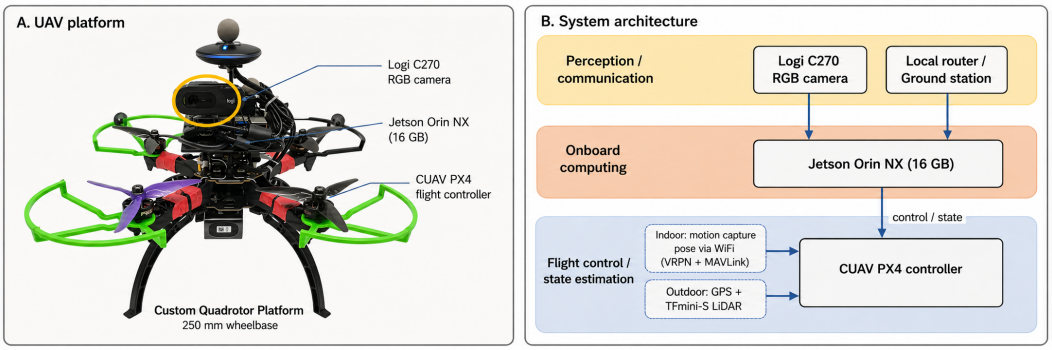


Figure 11: Real-world UAV platform and system architecture.

To evaluate the deployability of our proposed method in real-world environments, we develop a custom quadrotor platform with a 250 mm wheelbase. As shown in Figure 11, the platform is equipped with a Logi C270 RGB camera for egocentric visual perception and a Jetson Orin NX (16 GB) as the onboard computing unit, which is responsible for data reception, communication forwarding, and interface management with the low-level flight controller. It should be noted that high-level model inference is performed on the ground-station server, while the UAV mainly handles visual observation acquisition, control command reception, and flight execution. Low-level state estimation and flight control are managed by a CUAV PX4 flight controller operating under closed-loop position control. For reliable data transmission, the UAV connects wirelessly to a local router, which is further tethered to the ground-station server via a wired network, forming a real-time communication link between the UAV and the server.

We designed two real-world experimental scenarios, including indoor and outdoor settings, to comprehensively evaluate the system performance. The indoor experiments were conducted in a 10 m $\times$ 15 m $\times$ 3 m flight arena equipped with a 14-camera motion capture system, which provides highly accurate external pose estimation with sub-millimeter accuracy ($<$ 1 mm). The pose data from the motion capture system is streamed to the UAV over WiFi and integrated into the PX4 flight

controller through the VRPN package and the MAVLink protocol for closed-loop position control and trajectory recording.

The outdoor experiments were conducted in a relatively enclosed, open area with reliable GPS signal reception. To improve the robustness of outdoor state estimation, the GPS data is further complemented by a rigidly mounted Northwake TFmini-S LiDAR rangefinder, which provides accurate altitude estimation. Through the combined evaluation in the indoor motion-capture environment and the outdoor GPS/LiDAR environment, we verify the executability and environmental adaptability of the proposed method on a real UAV platform.

Importantly, neither the motion-capture poses in indoor experiments nor the GPS/LiDAR measurements in outdoor experiments are provided to the model as input; they are used only for low-level flight stabilization, safe motion execution, and trajectory recording, while WorldVLN makes high-level navigation decisions solely from egocentric RGB observations and language instructions.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The main claims in the abstract and introduction reflect the proposed autoregressive WAM architecture, the two-stage training framework, and the experimental scope reported in Sections 4–6.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss limitations in Section 6 and Appendix A.2, including short-range navigation, limited real-world evaluation scope, server-side inference, robustness under challenging environments, and the lack of extensive multi-seed experiments.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: This paper does not present formal theoretical results, theorems, or proofs.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: We provide the model architecture, training objectives, rollout and reward design, benchmark protocols, input-output formats, and implementation details in Sections 4–5 and Appendix A.3–A.9.

Guidelines:

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

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

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

   Answer: [Yes]

   Justification: We provide a project website with demos, code, and instructions for reproducing the main experimental results. The paper also describes the required benchmarks, training settings, and evaluation protocols in Section 5 and Appendix A.4–A.9.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification:We specify the datasets, evaluation metrics, benchmark protocols, model backbones, training stages, optimizer settings, learning rates, batch sizes, and training budgets in Section 5 and Appendix A.4–A.9.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: We report single-run results without error bars because repeated full training of the 8B autoregressive backbone and simulator-based rollout fine-tuning is computationally expensive. We discuss this limitation in Appendix A.2 and control the comparison using fixed test splits, consistent evaluation protocols, and the same action normalization where applicable.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We report the main training and evaluation hardware in Section 5 and provide additional training configurations in Appendix A.4 and Appendix A.8. The final supervised training run of the WAM backbone requires approximately 400 A800 GPU-hours, and simulator-based rollouts are executed on an RTX 4090 workstation.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and conducted the work in accordance with its principles, including transparency, reproducibility, and responsible discussion of potential deployment risks.

   Guidelines:

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

10. **Broader impacts**

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

Answer: [Yes]

Justification: We discuss potential positive applications and negative societal impacts of UAV-based language-conditioned navigation, including physical safety, privacy, surveillance, and misuse risks, in Appendix A.1.

Guidelines:

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

11. **Safeguards**

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

Answer: [Yes]

Justification: We describe responsible deployment safeguards in Appendix A.1, including controlled test environments, human supervision, geofencing, speed and altitude limits, emergency stop mechanisms, reliable state-estimation checks, and compliance with local UAV regulations.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: We cite the creators of the existing benchmarks, pretrained models, and software components used in this work, including UAV-Flow, IndoorUAV-VLA, InfinityStar-8B, the adopted video VAE, PX4, MAVLink, and VRPN, and summarize their usage and corresponding licenses or terms in Appendix A.5.

Guidelines:

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

13. **New assets**

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

Answer: [Yes]

Justification: We release the project code through an anonymized project website and provide documentation including usage instructions, training and evaluation details, license information, and limitations.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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

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

Answer: [N/A]

Justification: This work does not involve crowdsourcing or research with human subjects.

Guidelines:

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

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

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

    Answer: [N/A]

    Justification: This work does not involve human-subject research requiring IRB approval or equivalent institutional review.

    Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: The core method development of this research does not use LLMs as an important, original, or non-standard component. Any LLM usage is not part of the proposed WorldVLN method.

    Guidelines:

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