DRONEWAM: EFFICIENT WORLD ACTION MODEL FOR DRONE VISUAL NAVIGATION

Anonymous authors
Paper under double-blind review

ABSTRACT

World-action models give visual navigation agents a way to anticipate how candidate actions will change future observations and to act from the predicted consequences. For drones, this capability must operate under tight accuracy and efficiency constraints. We present DroneWAM, an efficient world-action model for drone visual navigation. DroneWAM adopts a JEPA-based architecture to model future states directly in representation space, avoiding the cost of explicit future image generation. A pretrained Resampler further compresses dense encoder features into fewer latent tokens, reducing the computation repeated at each imagined step. We also introduce adaptive rollout, where a preference-trained Gate adaptively allocates prediction depth according to the current scene. To support learning under richer aerial motion, we construct DroneNav-6D, a simulated visual navigation dataset with synchronized RGB observations, 6-DoF flight trajectories, control commands, and randomized wind disturbances. On DroneNav-6D, DroneWAM achieves the best trajectory accuracy among the compared methods. Adaptive rollout further reduces the average prediction depth from 8 to 4.58 while improving trajectory accuracy, demonstrating that predictive computation can be allocated more effectively across scenes.

1 INTRODUCTION

Visual navigation (Zhang et al., 2022; Nahavandi et al., 2025; Jiang et al., 2026) is a central capability for autonomous drones. A drone navigates from partial egocentric observations, while each control command changes both its physical state and future visual input. Therefore, reliable navigation requires anticipating how candidate actions may affect future observations and progress toward the goal. World Models (Matsuo et al., 2022; Chen et al., 2025) provide such predictive capability by modeling state transitions, while World-Action Models (WAMs) (Shen et al., 2026; Lu et al., 2026b;a) further connect future prediction with action generation. Since this prediction is repeatedly invoked throughout closed-loop flight under limited onboard computation, an aerial WAM should provide accurate foresight with high inference efficiency.

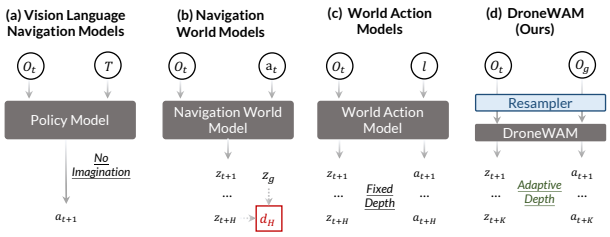


Figure 1: Compared to prior approaches, DroneWAM performs compact latent prediction with adaptive rollout.

Existing aerial navigation methods mainly approach this problem from two directions. As shown in Fig. 1, vision-language navigation methods (Liu et al., 2023b) condition action prediction on visual observations and semantic instructions, providing an effective interface for goal-directed navigation when language supervision is available. However, their future visual consequences are usually modeled only implicitly. Aerial world models and WAMs (Zhang et al., 2025; Zhao et al., 2026; Zheng et al., 2026b) instead explicitly predict how the visual world evolves with motion and use the predicted future to support action generation. Such predictive reasoning introduces additional computation during online control, since visual representations must be repeatedly propagated over multiple imagined steps. Moreover, existing models

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

commonly use a predefined rollout horizon, assigning similar predictive computation to different scenes.

These limitations suggest that efficiency should be considered directly in the predictive process of an aerial WAM. We identify three aspects that are particularly important. Firstly, the prediction space should retain the scene structure and motion information required for navigation without incurring the cost of unnecessary visual details. Secondly, the representation propagated through the world model should be compact, since its cost is repeatedly accumulated across rollout steps. Thirdly, the amount of prediction should depend on the current scene: straightforward situations may require only limited foresight, while ambiguous observations can benefit from deeper rollout. Together, these considerations motivate a world-action model that is both predictive and economical in how it represents and imagines the future.

Furthermore, while recent efforts have expanded the scale of video-action datasets (Chen et al., 2026a), training data for aerial world-action models should also capture the diverse translational and rotational motion of drones. Most established aerial visual-navigation benchmarks (Liu et al., 2023b; Zhu et al., 2026; Wang et al., 2025; Yao et al., 2025) represent flight with 4-DoF motion over translation and yaw. This abstraction provides limited coverage of roll and pitch, whose changes directly affect camera viewpoint and visual dynamics during flight. Higher-DoF visual-action data are therefore desirable for learning a more complete aerial world-action model.

Guided by these observations, we present DroneWAM, an efficient world-action model for drone visual navigation. DroneWAM adopts JEPA-based predictive learning (Assran et al., 2025; 2023; Chen et al., 2026b) to model future states directly in representation space, focusing computation on predictive scene structure and motion cues that are most relevant to navigation. A pretrained Resampler further compresses dense visual features into a compact token set, reducing the computation required at each imagined step. DroneWAM also adapts its rollout depth to the current scene through a preference-trained (Rafailov et al., 2023) Gate, allowing straightforward situations to use fewer prediction steps while allocating additional foresight to more ambiguous ones. These designs reduce both the spatial and temporal cost of world-action prediction while preserving the predictive information needed for navigation.

To support learning under richer aerial motion, we further construct DroneNav-6D, a simulated dataset for 6-DoF visual navigation. Since collecting large-scale real 6-DoF flight data is costly, we develop an automatic simulation pipeline that records synchronized RGB observations, flight commands, and 6-DoF trajectories across diverse environments, with randomized wind disturbances to broaden the covered flight conditions. DroneNav-6D provides more complete motion supervision for training world-action models and enables controlled evaluation under full 6-DoF aerial motion.

We evaluate DroneWAM on DroneNav-6D under both open-loop prediction and closed-loop navigation. In open-loop evaluation, DroneWAM achieves the lowest trajectory errors among all compared methods, reducing ATE by 39.8% and inference latency by 32.2% compared with FastWAM. In closed-loop evaluation, it improves navigation progress from 0.510 to 0.774 while maintaining lower trajectory error and inference cost. Ablation studies further validate the efficiency-oriented design: compressing the visual representation to 128 tokens reduces inference time by 18.3% compared with a 512-token representation with nearly unchanged trajectory accuracy, while adaptive rollout reduces the average prediction depth from 8 to 4.58 and lowers latency by 22.4% over fixed-depth rollout while achieving lower trajectory errors. We further observe similar benefits on LIBERO (Liu et al., 2023a), where adaptive rollout reduces inference latency by 53.8% relative to fixed-depth prediction with slightly improved action prediction accuracy. Our contributions are as follows:

- We introduce DroneWAM, an efficient world-action model for drone visual navigation. It provides a practical modeling framework for predictive navigation under constrained onboard computation.
- We construct DroneNav-6D, a simulated 6-DoF aerial navigation dataset built with an automatic data generation pipeline. It offers a scalable resource for training and evaluating aerial world-action models beyond conventional 4-DoF motion.
- Extensive experiments demonstrate that DroneWAM achieves both higher navigation accuracy and lower inference cost than existing world-model and world-action baselines.

2 DRONENAV-6D

To support predictive learning with translational and attitude changes, we develop an automatic data collection pipeline using Unreal Engine 4.27 and AirSim (Shah et al., 2017). AirSim provides multirotor simulation, RGB rendering, and synchronized state recording. We collect data in five simulation worlds: traffic roads in SnappyRoads, snowy mountains, a coastal city in CITYBIM, rural and agricultural areas in RuralAustralia, and dense urban streets in NYC. These worlds provide varied scene layouts, terrain, and visual appearances.

For each flight, a multirotor equipped with a forward-facing RGB camera ascends to a predefined cruising altitude. We sample waypoints within scene-specific flight boundaries by varying the travel direction, displacement, and altitude offset. The waypoints are connected into a smooth three-dimensional trajectory and executed through velocity and heading commands. During execution, we record RGB observations together with the vehicle position, orientation, and issued commands. Consecutive records associate visual transitions with commanded inputs and realized pose changes, providing supervision for future prediction.

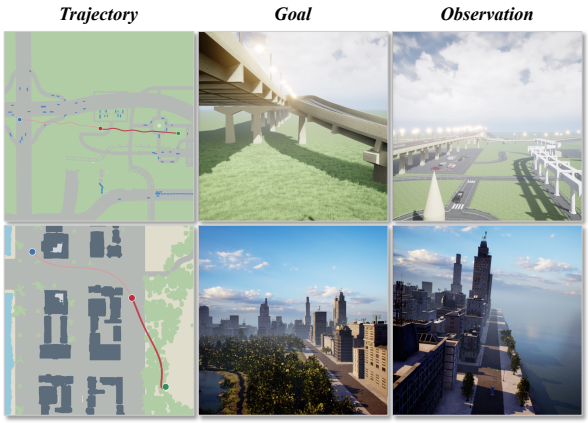


Figure 2: Examples of goal-conditioned navigation samples in DroneNav-6D. Each row shows the flight trajectory together with the sampled goal image and current observation.

We inject random wind disturbances during flight and record the resulting motion, including lateral drift, attitude changes, and controller corrections. The paired command and pose records capture the commanded inputs and realized motion under these disturbances. Episodes are filtered using checks on collisions, prolonged immobility, trajectory length, frame intervals, and action validity. This procedure yields temporally ordered visual and motion sequences for predictive learning.

The resulting DroneNav-6D corpus contains 3,837 trajectory clips with 1,592,170 RGB observations, totaling 63.03 hours of recorded flight. The median recording rate is approximately 7.00 Hz. Training and validation contain 3,453 and 384 clips, respectively, with distinct clip identifiers and all five worlds represented in both splits.

To support image-goal navigation, we further construct goal-conditioned samples from the recorded flight trajectories. For each sample, the current observation and a future goal image are selected from the same trajectory, while the synchronized intermediate states and actions provide the corresponding motion supervision. As illustrated in Fig. 2, we show two representative navigation samples from DroneNav-6D. The first corresponds to a relatively open environment with an approximately straight navigation trajectory, while the second is collected in a dense urban area where the drone needs to turn between buildings, representing a more complex navigation case. Further dataset statistics and analyses are provided in Appendix D.

3 DRONEWAM

3.1 PROBLEM SETUP

Following prior work on image-goal navigation (Shah et al., 2023; Sridhar et al., 2024), we specify the navigation target with a goal image g. At control step t, the drone receives an RGB observation o_t and a motion state s_t, and has access to its executed flight commands $a_{\leq t}$. The observation and goal image are processed by the same visual encoder E and Resampler R_ρ, yielding $z_t = R_\rho(E(o_t))$ and $z_g = R_\rho(E(g))$, respectively. Given the observed context and the goal latent z_g, a world-action model predicts future latent states and an H-step action plan for navigation. We use k to denote

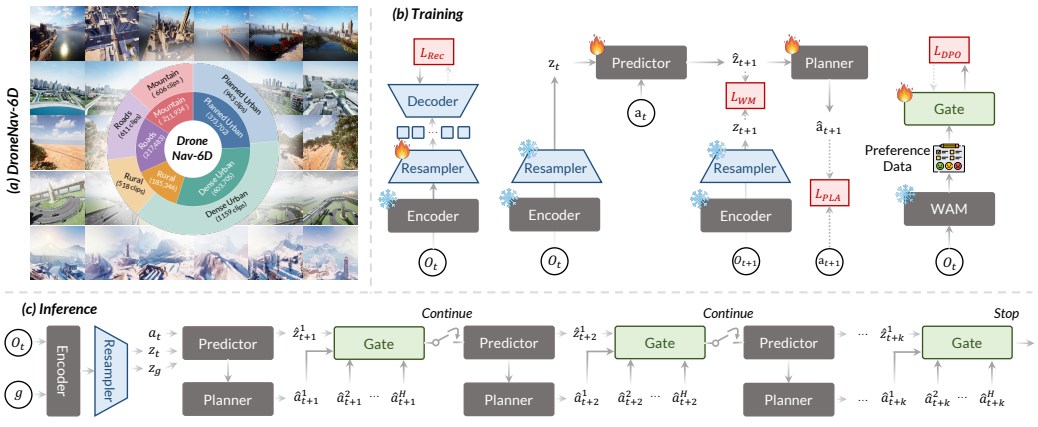


Figure 3: Overview of DroneWAM and DroneNav-6D. (a) DroneNav-6D provides diverse simulated environments with synchronized visual observations and 6-DoF flight trajectories. (b) DroneWAM is trained in three stages: Resampler pretraining, world-action model training with latent prediction and planning objectives, and preference learning for the Gate. (c) During inference, the Predictor and Planner iteratively imagine future latent states and action plans, while the Gate determines when to stop and returns the highest-scoring plan for efficiency.

the number of imagined world transitions and h to denote the temporal index within an action plan. Thus, the rollout depth and the action horizon describe two different temporal dimensions.

Conventional WAMs perform a predefined number of world transitions K for each control update. We instead formulate world-action prediction as a variable-length rollout:

$$\tau_t = \min\left(\{k \mid d_{t,k} = \text{stop}\} \cup \{K_{\max}\}\right), \tag{1}$$

where $d_{t,k}$ is the stopping decision after the k-th imagined transition. At each depth k, the model produces a predicted latent state $\hat{z}_{t+k}$ and an H-step action plan $\hat{\mathbf{A}}_t^{(k)}$. The resulting depth τ_t therefore controls the amount of world prediction performed for the current scene, while H remains the planning horizon of each candidate plan.

3.2 Training

Resampler pretraining. Given an RGB observation o_t, a frozen visual encoder E produces dense visual tokens $x_t = E(o_t) \in \mathbb{R}^{M \times d}$. We pretrain a lightweight Resampler R_ρ to compress them into a fixed set of $N = 128$ latent tokens, $z_t = R_\rho(x_t)$, while a lightweight Decoder reconstructs the original encoder features from the compressed representation. The Resampler and Decoder are trained with feature reconstruction and temporal consistency objectives. This produces a compact visual representation that preserves the encoder information while reducing the computation of subsequent multi-step prediction. After pretraining, the encoder and Resampler are frozen, and the Decoder is discarded. The goal image is encoded using the same frozen encoder and Resampler to obtain the goal latent $z_g = R_\rho(E(g))$.

Predictor and Planner training. Let $C_t = \{z_{\le t}, s_{\le t}, a_{\le t}\}$ denote the observed visual, motion-state, and executed-action context. Conditioned on C_t and the goal latent z_g, the Predictor F_θ autoregressively predicts up to $K_{\max}$ future latent states:

$$\hat{z}_{t+k} = F_\theta\left(C_t, z_g, \hat{z}_{t+1:t+k-1}\right), \qquad k = 1, \ldots, K_{\max}. \tag{2}$$

Future ground-truth states and actions are masked from the Predictor input during training. The Predictor therefore uses only the observed context, the goal latent, and previously predicted latent states. We denote the target representation at depth k by $z^\star_{t+k}$ and supervise world prediction with a temporally discounted latent loss:

$$\mathcal{L}_{\text{WM}} = \frac{\sum_{k=1}^{K_{\max}} \gamma^{k-1} \left\| \hat{z}_{t+k} - z^\star_{t+k} \right\|_1}{\sum_{k=1}^{K_{\max}} \gamma^{k-1}}. \tag{3}$$

At each imagination depth k, the Planner P_ψ takes the predicted latent prefix and produces a complete H-step 6-DoF action plan (Zhao et al., 2023) :

$$\hat{\mathbf{A}}_t^{(k)} = P_\psi\left(\hat{z}_{t+1:t+k}, z_g\right) = \left(\hat{a}_{t+1|t}^{(k)}, \ldots, \hat{a}_{t+H|t}^{(k)}\right). \tag{4}$$

Here, k denotes the world imagination depth, while $h \in \{1, \ldots, H\}$ indexes the temporal position within the action plan. Thus, the H outputs form one future action sequence rather than H candidate actions at the same time step.

Each action plan is integrated from the current 6-DoF pose to obtain a predicted trajectory. We supervise the Planner at every imagination depth using relative absolute trajectory error (rATE) and relative pose error (rRPE):

$$\begin{aligned} \mathcal{L}_{\mathrm{PLA}} &= \frac{1}{K_{\max}} \sum_{k=1}^{K_{\max}} \left(\frac{\mathrm{rATE}^{(k)}}{\epsilon_A} + \frac{\mathrm{rRPE}^{(k)}}{\epsilon_R} \right), \\ \mathcal{L}_{\mathrm{WAM}} &= \lambda_{\mathrm{WM}} \mathcal{L}_{\mathrm{WM}} + \lambda_{\mathrm{PLA}} \mathcal{L}_{\mathrm{PLA}}. \end{aligned} \tag{5}$$

Only the Predictor and Planner are optimized in this stage, while the representation modules and target branch remain frozen.

Gate preference training. After WAM training, we freeze the complete WAM and evaluate every imagination depth $k \in \{1, \ldots, K_{\max}\}$ for each sample i. Each depth produces an H-step action plan and its corresponding trajectory errors. We define a cost that jointly considers planning quality and rollout computation:

$$C_{i,k} = \frac{\mathrm{rATE}_{i,k}}{\epsilon_A} + \frac{\mathrm{rRPE}_{i,k}}{\epsilon_R} + \lambda_c \frac{k}{K_{\max}}. \tag{6}$$

The depth with the lowest cost is treated as the preferred sample, while the remaining depths form rejected samples.

The Gate input $u_{i,k}$ contains only information available at inference time, including the observed latent state, predicted latent history, action-plan history, changes between adjacent predictions, recent 6-DoF motion, current velocity, and the imagination depth. The Gate predicts low-reward and high-reward logits. Given a fixed reference Gate, we define a DPO-style reference-adjusted reward and optimize the Gate:

$$\begin{aligned} r_\omega(u) &= \log \frac{p_\omega(\mathrm{high} \mid u)}{p_\omega(\mathrm{low} \mid u)} - \log \frac{p_{\mathrm{ref}}(\mathrm{high} \mid u)}{p_{\mathrm{ref}}(\mathrm{low} \mid u)}, \\ \mathcal{L}_{\mathrm{DPO}} &= -\mathbb{E}\left[\log \sigma\left(\beta\left[r_\omega(u^+) - r_\omega(u^-)\right]\right)\right]. \end{aligned} \tag{7}$$

Only the Gate is updated during this stage. We then use a disjoint calibration subset to determine a stopping threshold η_k for each imagination depth.

3.3 Adaptive Inference

At control step t, the shared frozen Encoder and Resampler encode the observed images and the goal image into latent representations. Conditioned on the observed context C_t and the goal latent z_g, the model then performs world prediction, planning, and Gate evaluation iteratively. At depth k, the Predictor generates $\hat{z}_{t+k}$, the Planner produces $\hat{\mathbf{A}}_t^{(k)}$, and the Gate assigns the current plan a high-reward score $q_{t,k}$. The rollout stops once the score exceeds the calibrated threshold:

$$\begin{aligned} \tau_t &= \min\left(\{k \mid K_{\min} \le k \le K_{\max},\ q_{t,k} \ge \eta_k\} \cup \{K_{\max}\}\right), \\ \kappa_t &= \arg\max_{1 \le k \le \tau_t} q_{t,k}. \end{aligned} \tag{8}$$

The stopping depth τ_t determines the number of world-prediction steps actually performed, while κ_t determines which action plan is finally selected. The two may differ because the model retains the highest-scoring plan encountered before termination. The controller executes only the first action of $\hat{\mathbf{A}}_t^{(\kappa_t)}$. After receiving the next observation, the entire procedure is repeated in a receding-horizon manner. The goal image is provided as task input.

Table 1: Open-loop evaluation on DroneNav-6D.

Method	ATE↓	Rel.ATE↓	RPE↓	Rel.RPE↓	Infer. (ms)↓
DINO-WM (Zhou et al., 2024)	9.0923	0.4459	3.0681	0.1502	29,622.24
V-JEPA 2-AC (Assran et al., 2025)	27.4317	0.6621	9.0439	0.2190	64,617.72
LeWM (Maes et al., 2026)	4.2259	0.4164	1.0924	0.1080	654.37
FastWAM (Yuan et al., 2026)	1.8366	0.1919	0.5086	0.0427	712.33
NWM (Bar et al., 2025)	3.7580	0.3740	0.8051	0.0802	403,827.78
Ours	**1.1062**	**0.0578**	**0.3470**	**0.0180**	**483.29**

Table 2: Closed-loop navigation results on DroneNav-6D.

Method	ATE↓	Rel.ATE↓	RPE↓	Rel.RPE↓	Prog.↑	MSE↓	Infer. (ms)↓
FastWAM (Yuan et al., 2026)	119.8063	0.7410	2.7309	0.0176	0.510	124.30	712.33
NWM (Bar et al., 2025)	88.6305	0.5980	2.7245	0.0171	0.500	178.55	403,827.78
Ours	**83.4461**	**0.5184**	**2.7194**	**0.0168**	**0.774**	**91.96**	**483.29**

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Implementation Details. DroneWAM predicts an eight-step ego-frame 6-DoF action plan from four RGB observations, associated 7-D motion states, executed-action history, and a goal image. Observations and goals share a frozen ViT-L encoder and pretrained TokenAE resampler ($512 \rightarrow 128$ tokens), with goal latents supplied at every imagination depth. The Predictor has 24 Transformer layers of width 1024, and the Planner maintains a 256-D recurrent state. Base WAM training uses 30 epochs of AdamW with bfloat16, a peak learning rate of 7.5×10^{-5}, and an effective batch size of 128, followed by five epochs of variable-depth Predictor–Planner adaptation. The five-member marginal-value Gate is trained for 30 epochs. Model training uses $8 \times$ A100-80GB GPUs.

Metrics. We report Absolute Trajectory Error (ATE), Relative Pose Error (RPE), and their trajectory-length-normalized variants, rel. ATE and rel. RPE. We additionally report synchronized inference latency and average autoregressive rollout depth. Closed-loop evaluation further includes task progress, and MSE.

4.2 MAIN RESULTS

Open-Loop Results. We first evaluate prediction accuracy and inference efficiency under open-loop rollout, where all methods are given the same observation context and predict future motion without intermediate feedback. As shown in Tab. 1, DroneWAM achieves the best performance across all trajectory metrics while also requiring the lowest inference latency. Compared with FastWAM, DroneWAM reduces ATE from 1.8366 to 1.1062 and Rel.ATE from 0.1919 to 0.0578, while reducing inference time from 712.33 ms to 483.29 ms. These results show that DroneWAM achieves a better accuracy–efficiency trade-off than the compared baselines, with lower trajectory error and lower inference latency.

Closed-Loop Results. We further evaluate closed-loop navigation in Tab. 2, where the model replans from new observations. DroneWAM achieves the highest progress of 0.774, the lowest trajectory errors and MSE, and lower inference latency than the compared methods. However, none of the evaluated methods completes the task, so the current system is not yet ready for real-world deployment. These gains nevertheless suggest that efficient world-action modeling is a promising direction for practical drone navigation. Additional closed-loop evaluations are provided in Appendix F.

4.3 ABLATION STUDIES

4.3.1 COMPONENT ABLATION

We progressively introduce the major components to examine their contributions to prediction accuracy and efficiency. The planner provides the main improvement in trajectory accuracy. Adding the Resampler reduces inference time from 656.12 ms to 582.65 ms with only marginal changes in trajectory error, showing that dense encoder features contain substantial redundancy for world prediction. Adaptive Rollout further reduces latency to 483.29 ms while improving both Rel.ATE and Rel.RPE. The Resampler and Adaptive Rollout reduce computation from complementary spatial and temporal perspectives.

Table 3: Component ablation. PLA, RS, and AR denote Planner, Resampler, and Adaptive Rollout, respectively.

PLA	RS	AR	Rel.ATE↓	Rel.RPE↓	Infer. (ms)↓
			0.6621	0.2190	64,617.72
✓			0.0584	0.0189	656.12
✓	✓		0.0603	0.0192	582.65
✓	✓	✓	**0.0578**	**0.0180**	**483.29**

4.3.2 Adaptive Rollout Strategy

We compare different stopping strategies to determine whether the gain arises from early termination itself or from selecting an appropriate rollout depth. Our method reduces the average rollout from 8.0 to 4.578 while achieving lower trajectory errors than fixed rollout. More importantly, random stopping uses nearly identical computation with an average depth of 4.581, yet performs considerably worse. This shows that the learned Gate improves the allocation of predictive computation rather than simply shortening the rollout.

Table 4: Comparison of rollout strategies. Avg. K denotes the average rollout depth.

Method	Rel.ATE↓	Rel.RPE↓	Avg. K↓
Fixed	0.0688	0.0221	8.000
Random	0.0722	0.0224	4.581
Latent Margin	0.0887	0.0309	5.625
Ours	**0.0578**	**0.0180**	**4.578**

4.3.3 Number of Resampler Tokens

We vary the number of Resampler tokens to study the trade-off between representation capacity and inference efficiency. Compressing 512 tokens to 128 reduces latency from 591.28 ms to 483.29 ms with almost unchanged Rel.ATE and Rel.RPE. This yields an 18.3% latency reduction with absolute error increases of only 0.0004 and 0.0001, respectively. Further compression to 16 tokens reduces latency again but causes a clear degradation in trajectory accuracy. We therefore use 128 tokens as the default configuration, providing a favorable balance between compactness and predictive fidelity.

Table 5: Effect of Resampler size. #Tok. denotes the number of output latent tokens.

#Tok.	Rel.ATE↓	Rel.RPE↓	Infer. (ms)↓
512	**0.0574**	**0.0179**	591.28
128	0.0578	0.0180	483.29
16	0.0659	0.0198	**405.16**

4.4 Further Analysis

Why Does Adaptive Rollout Improve Prediction Accuracy? We evaluate every sample at all rollout depths from $K = 0$ to $K = 8$ and group it by the depth that achieves the lowest trajectory error. As shown in Fig. 4, the optimal depths vary substantially across samples rather than concentrating at the maximum rollout depth. Some samples are best predicted with only shallow imagination, while others continue to benefit from deeper rollout, indicating that more prediction is not always better. It suggests that the useful amount of imagination depends on the current scene. Adaptive rollout can therefore improve both accuracy and efficiency by learning to allocate an appropriate prediction depth to each input.

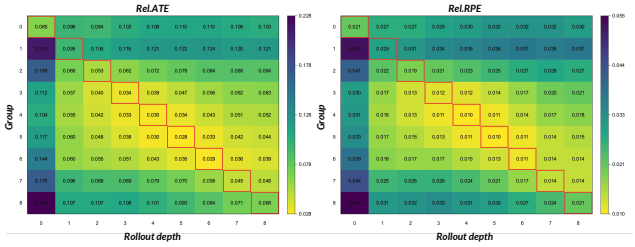


Figure 4: Accuracy at different rollout depths. Samples are grouped by the rollout depth, and red boxes indicate the corresponding selected depth.

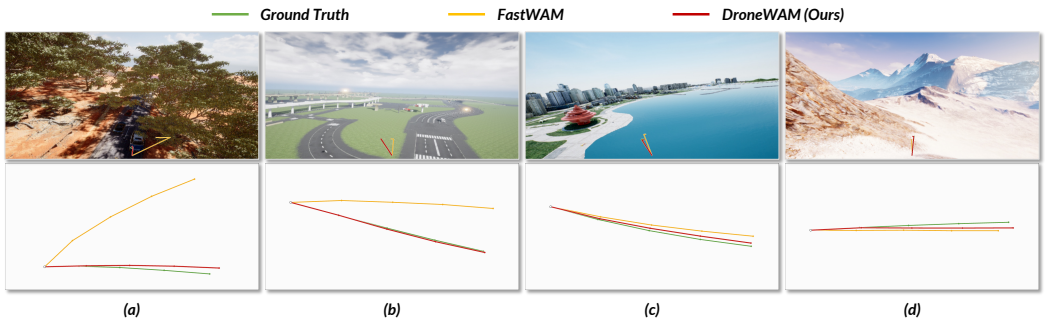


Figure 6: Qualitative comparison of predicted trajectories across diverse aerial scenes. Green, yellow, and red denote the ground-truth trajectory, FastWAM, and DroneWAM, respectively. DroneWAM consistently follows the reference trajectory more closely, while FastWAM exhibits larger directional and accumulated drift.

Does Adaptive Rollout Generalize Beyond Aerial Navigation? To examine whether adaptive computation is specific to aerial navigation, we apply the same rollout strategy to LIBERO robotic manipulation. Adaptive rollout reduces inference latency from 890.8 ms to 411.9 ms, a 53.8% reduction, while slightly improving normalized action error and gripper accuracy. The remaining action-error metrics show the same trend and are reported in the Appendix. Moreover, the improvement over fixed $K = 8$ holds across all four LIBERO suites, while the average selected depth varies across task families. These results suggest that adaptive rollout transfers beyond aerial navigation and adjusts its computation to different task distributions.

Table 6: Open-loop results on LIBERO. Full action metrics are reported in the Appendix.

Method	N.RMSE↓	Grip.(%)↑	Latency↓
$K = 8$	0.2685	89.83	890.8
Adaptive	**0.2652**	**90.35**	**411.9**

Can the Preference-Trained Gate Generalize to Unseen Environments? We further examine whether preference learning causes the Gate to overfit to environments observed during training. We use environment-level cross-validation, defining environments included in Gate preference training as seen and those held out from this training as unseen. As shown in Fig. 5, the seen and unseen results remain closely aligned for both Rel.ATE and Rel.RPE across all five environments. The maximum absolute gap is only 0.0034 for Rel.ATE and 0.0010 for Rel.RPE, and unseen errors are lower in seven of the ten environment–metric comparisons. Importantly, each unseen result is obtained without preference pairs from the corresponding environment during Gate training. This indicates that our Gate does not exhibit evident overfitting to the environments used during training.

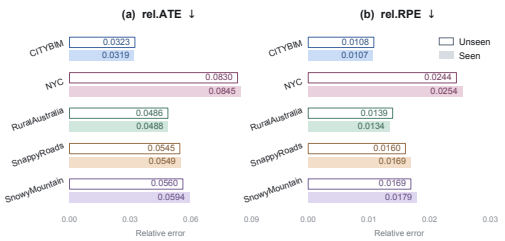


Figure 5: Generalization of the preference-trained Gate across unseen environments. Seen and unseen results are compared under environment-level cross-validation.

Qualitative Results. We qualitatively examine whether the improvements in trajectory metrics translate into more faithful motion prediction across different environments. Fig. 6 compares the predicted trajectories of DroneWAM and FastWAM in four representative scenes. DroneWAM consistently remains closer to the ground-truth trajectory, preserving both the motion direction and trajectory shape more accurately. The difference is particularly evident when longer-term prediction amplifies small motion errors, where FastWAM exhibits substantial directional drift, while DroneWAM maintains a more stable trajectory. These examples complement the quantitative results, showing that the lower trajectory errors of DroneWAM correspond to more consistent motion prediction across diverse aerial environments.

5 Related Work

5.1 World Models and World-Action Models

World models Azuma et al. (2026); Zhang & Du (2026); Hafner et al. (2019); Ha & Schmidhuber (2018) learn predictive dynamics that allow an agent to reason about the consequences of its actions before execution. Early model-based agents Wang et al. (2026); Chahe & Zhou (2026) typically learn compact latent dynamics and perform policy optimization through imagined trajectories, as exemplified by DreamerV3 (Hafner et al., 2025). Recent works increasingly move visual prediction into representation space. V-JEPA 2 (Assran et al., 2025) and DINO-WM (Zhou et al., 2024) predict action-conditioned future features and support planning without explicit RGB reconstruction, substantially reducing the complexity of modeling visual dynamics. World-Action Models further couple world prediction with action generation. DreamZero (Ye et al., 2026) jointly models future visual states and actions, while WAM-Nav (Yang et al., 2026) uses asymmetric latent world-action modeling to balance visual foresight and action planning. These works establish predictive representations as a useful interface between perception and control. DroneWAM follows this direction while focusing on the inference cost of recurrent prediction.

5.2 Drone Navigation

Drone navigation Yan et al. (2025); Zheng et al. (2026a); Sun et al. (2026) has been studied under both visual goal conditioning and language-guided settings. Visual navigation methods use images, geometric goals, or predicted future observations to evaluate motion and select trajectories. Navigation World Models (Bar et al., 2025) demonstrate trajectory evaluation through controllable future prediction, while recent aerial extensions model long-range visual dynamics in three-dimensional environments (Zhang et al., 2025; Zhu et al., 2026). Aerial vision-language navigation instead grounds natural-language instructions in egocentric observations. AerialVLN (Liu et al., 2023b) established an outdoor UAV VLN setting, and subsequent benchmarks such as OpenUAV (Wang et al., 2025), CityNav (Lee et al., 2025), and OpenFly (Gao et al., 2026) expand the scale, realism, and diversity of aerial navigation data. More recent approaches integrate predictive world modeling into aerial VLN. WorldVLN (Zhao et al., 2026) performs autoregressive latent world-action prediction, while WorldFly (Zheng et al., 2026b) jointly generates future visual states and navigation actions. These developments bring explicit foresight to aerial navigation, although online prediction remains computationally demanding and most available navigation data still simplify aerial motion. Our work studies efficient world-action prediction together with higher-DoF aerial supervision.

6 Conclusion

We present DroneWAM, an efficient world-action model for drone visual navigation. DroneWAM combines JEPA-based latent prediction, compact visual representations, and adaptive rollout to reduce the cost of future imagination while maintaining strong navigation accuracy. We further introduce DroneNav-6D, a simulated dataset with synchronized visual observations and full 6-DoF aerial motion. Experiments demonstrate consistent gains in both prediction quality and inference efficiency, and show that adaptive rollout can allocate computation effectively across different scenes. We hope it can support compact and adaptive predictive modeling as a practical direction for autonomous drone navigation.

7 Limitations

DroneWAM's fixed token budget and bounded imagination horizon may require adjustment across scenes and computational budgets. Performance also depends on the information retained by its pretrained encoder and resampler. In simulation, closed-loop task success remains very low in complex scenes, leaving a substantial gap to reliable navigation. The current system is therefore not ready for real-world deployment. Practical use requires higher success rates, improved robustness, and validation on physical platforms. But, we believe compact world-action modeling with adaptive prediction remains a promising direction, with better representations, broader training coverage, and improved control integration offering a path toward reliable drone autonomy.

REFERENCES

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15619–15629. IEEE, 2023.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

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

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

Amirhosein Chahe and Lifeng Zhou. Policy-guided world model planning for language-conditioned visual navigation, 2026. URL https://arxiv.org/abs/2603.25981.

Delong Chen, Theo Moutakanni, Willy Chung, Yejin Bang, Ziwei Ji, Allen Bolourchi, and Pascale Fung. Planning with reasoning using vision language world model, 2025. URL https://arxiv.org/abs/2509.02722.

Delong Chen, Tejaswi Kasarla, Yejin Bang, Mustafa Shukor, Willy Chung, Jade Yu, Allen Bolourchi, Theo Moutakanni, and Pascale Fung. Action100m: A large-scale video action dataset. *arXiv preprint arXiv:2601.10592*, 2026a.

Delong Chen, Mustafa Shukor, Théo Moutakanni, Willy Chung, Lei Yu, Tejaswi Kasarla, Allen Bolourchi, Yann LeCun, and Pascale Fung. Vl-jepa: Joint embedding predictive architecture for vision-language. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 89791–89807, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/9144aded4e536bc5fb7bfc660a2d7a3d-Paper-Conference.pdf.

Yunpeng Gao, Chenhui Li, Zhongrui You, Junli Liu, Li Zhen, Pengan Chen, Qizhi Chen, Zhonghan Tang, Liansheng Wang, Yiwen Tang, et al. Openfly: A comprehensive platform for aerial vision-language navigation. In *International Conference on Learning Representations*, volume 2026, pp. 101858–101871, 2026.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3):440, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

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

Jianyu Jiang, Zequan Wang, Liang Yao, Shengxiang Xu, and Fan Liu. Airnavigation: Let uav navigation tell its own story. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 41610–41612, 2026.

Jungdae Lee, Taiki Miyanishi, Shuhei Kurita, Koya Sakamoto, Daichi Azuma, Yutaka Matsuo, and Nakamasa Inoue. Citynav: A large-scale dataset for real-world aerial navigation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5912–5922. IEEE, 2025.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 44776–44791. Curran Associates, Inc., 2023a. doi:

10.52202/075280-1939. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/8c3c666820ea055a77726d66fc7d447f-Paper-Datasets_and_Benchmarks.pdf.

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)*, pp. 15338–15348. IEEE, 2023b.

Hongbo Lu, Liang Yao, Chenghao He, Hao Han, Fan Liu, Wenlong Liao, Tao He, and Pai Peng. Rise: Adaptive imagination for world action models. *arXiv preprint arXiv:2608.20430*, 2026a.

Hongbo Lu, Liang Yao, Chenghao He, Haoyu Wang, Xiang Gu, Xianfei Li, Wenlong Liao, Tao He, and Pai Peng. The dawn of world-action interactive models. *arXiv preprint arXiv:2605.11550*, 2026b.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworld-model: Stable end-to-end joint-embedding predictive architecture from pixels, 2026. URL https://arxiv.org/abs/2603.19312.

Yutaka Matsuo, Yann LeCun, Maneesh Sahani, Doina Precup, David Silver, Masashi Sugiyama, Eiji Uchibe, and Jun Morimoto. Deep learning, reinforcement learning, and world models. *Neural networks*, 152:267–275, 2022.

Saeid Nahavandi, Roohallah Alizadehsani, Darius Nahavandi, Shady Mohamed, Navid Mohajer, Mohammad Rokonuzzaman, and Ibrahim Hossain. A comprehensive review on autonomous navigation. *ACM Computing Surveys*, 57(9):1–67, 2025.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Dhruv Shah, Ajay Sridhar, Nitish Dashora, Kyle Stachowicz, Kevin Black, Noriaki Hirose, and Sergey Levine. Vint: A foundation model for visual navigation. *arXiv preprint arXiv:2306.14846*, 2023.

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

Qiuhong Shen, Shihua Zhang, Yue Liao, Qi Li, Zhenxiong Tan, Shizun Wang, Shuicheng Yan, and Xinchao Wang. World action models: A survey. *arXiv preprint arXiv:2606.20781*, 2026.

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

Xiaolou Sun, Wufei Si, Wenhui Ni, Yuntian Li, Dongming Wu, Fei Xie, Runwei Guan, He-Yang Xu, Henghui Ding, Yuan Wu, Yutao Yue, Yongming Huang, and Hui Xiong. Autofly: Vision-language-action model for UAV autonomous navigation in the wild. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=88RKxlFUNY.

Xiangyu Wang, Donglin Yang, Ziqin Wang, Hohin Kwan, Jinyu Chen, Hongsheng Li, Yue Liao, Si Liu, et al. Towards realistic uav vision-language navigation: Platform, benchmark, and methodology. In *International Conference on Learning Representations*, volume 2025, pp. 7292–7310, 2025.

Yucen Wang, Fengming Zhang, De-Chuan Zhan, Li Zhao, Kaixin Wang, and Jiang Bian. Co-evolving latent action world models, 2026. URL https://arxiv.org/abs/2510.26433.

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

Ning Yang, Yan Huang, Kaiwen Peng, Ziheng He, Kai Wang, Cui Miao, Kailin Lyu, Guo Li, Xiaofeng Wang, Zheng Zhu, et al. Wam-nav: Asymmetric latent world-action modeling for unified visual navigation. *arXiv preprint arXiv:2606.04907*, 2026.

Liang Yao, Fan Liu, Shengxiang Xu, Chuanyi Zhang, Shimin Di, Xing Ma, Jianyu Jiang, Zequan Wang, and Jun Zhou. Uemm-air: Enable uavs to undertake more multi-modal tasks. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 12792–12798, 2025.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026.

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

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

Xiangcheng Zhang and Yilun Du. World action planner: Generalizable decision-making with action-conditioned world models. *arXiv preprint arXiv:2607.27599*, 2026.

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

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Guiyong Zheng, Yueting Ban, Mingjie Zhang, Juepeng Zheng, and Boyu Zhou. Onfly: Onboard zero-shot aerial vision-language navigation toward safety and efficiency. *arXiv preprint arXiv:2603.10682*, 2026a.

Shengtao Zheng, Kai Li, Weichen Zhang, Yu Meng, Chen Gao, Xinlei Chen, Yong Li, and Xiao-Ping Zhang. Worldfly: A world-model-based vision-language-action model for uav navigation. *arXiv preprint arXiv:2606.06147*, 2026b.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

Deyi Zhu, Haoyu Fan, Yinan Zhu, Weichen Zhang, Shilin Ma, Xinlei Chen, and Yansong Tang. Uncertainty-aware world model for aerial image-goal navigation. *arXiv preprint arXiv:2608.05597*, 2026.

A STATEMENTS

A.1 AI USE STATEMENT

In this work, we used generative AI tools to assist with method implementation through code-writing support, polish the manuscript for readability, search for information, and identify relevant literature. We reviewed and verified all AI-assisted work. AI-assisted code was manually inspected and tested for correctness, and relevant literature identified with AI assistance was checked against the original sources. We take responsibility for the final content of this work, including all text, claims, code, and other artifacts produced with the aid of generative AI.

A.2 ETHICS STATEMENT

Our experiments are conducted in simulation, and DroneNav-6D contains rendered visual observations and simulator-recorded motion data. The low closed-loop success rates in complex scenes highlight safety challenges that require further investigation before physical deployment. Future deployment requires validation on real platforms, collision-prevention measures, and appropriate human supervision. Applications of drone navigation should protect personal privacy and follow applicable safety requirements. Any distribution of the dataset or associated simulation assets should respect the licenses of the underlying third-party resources.

A.3 REPRODUCIBILITY STATEMENT

The method section describes the model architecture, learning objectives, and training procedures. The experimental section reports implementation settings, computing resources, evaluation protocols, and ablation configurations. Section 2 documents the DroneNav-6D collection pipeline, including trajectory generation, synchronized recording, wind perturbations, and quality filtering. Appendix D provides dataset composition and split statistics, motion distributions, and the coordinate conventions used in the analysis.

B DETAILED EXPERIMENTAL SETTINGS

Data and preprocessing. Training samples are constructed from 20-frame video clips recorded at 7 Hz. The first four RGB frames and their associated motion states are used as observations, while the model predicts an eight-step action plan. Each motion state contains 3-D position, roll, pitch, yaw, and scalar speed. Actions contain ego-frame translation and rotation increments. Only observed states and previously executed actions are provided to the model; future states and actions are masked. Images are vertically cropped by 28 pixels at both boundaries, resized to 256×512, and normalized using ImageNet statistics.

Model configuration. The visual backbone is a frozen ViT-L with a patch size of 16, embedding width 1024, causal attention, and rotary positional embeddings. A pretrained TokenAE with four encoder layers, two decoder layers, and 16 attention heads reduces the number of visual tokens from 512 to 128. The TokenAE is frozen throughout training. The Predictor consists of 24 Transformer layers with width 1024, 16 attention heads, and an MLP expansion ratio of four. The Planner maintains a 256-D recurrent state and predicts an eight-step, 6-DoF action sequence. The goal-image representation is supplied to the Planner at every imagination depth.

Base training and adaptation. The base WAM is trained for 30 epochs using AdamW with bfloat16 precision, weight decay 0.04, and gradient clipping at 1.0. The learning rate is linearly increased from 10^{-6} to 7.5×10^{-5} during the first two epochs and subsequently cosine-decayed to 10^{-6}. We use a per-GPU batch size of 16 on eight A100-80GB GPUs, resulting in an effective batch size of 128. The visual encoder and TokenAE remain frozen, while the Predictor and action head are optimized. The action-regression loss is weighted by 0.35, and the autoregressive latent loss uses a temporal discount factor of 0.95. Translation dimensions receive larger action-loss weights than rotation dimensions.

After base training, the Predictor and Planner are jointly adapted for five epochs using rollouts of different imagination depths. The Predictor and Planner learning rates are 5×10^{-8} and 2×10^{-6}, respectively. Losses are averaged over all evaluated rollout depths, while the visual encoder, TokenAE, and action head remain frozen.

Gate training. The marginal-value Gate contains five independently initialized ensemble members, each implemented as an MLP with hidden width 128. Gate inputs include the 256-D Planner state, the current action plan, changes in the Planner state and action plan from the previous depth, and a learned depth embedding. Each member is trained for 30 epochs using AdamW with a batch size of 512, learning rate 10^{-4}, weight decay 0.04, and gradient clipping at 1.0. Bootstrap sampling is performed at the trajectory level, and 20% of the training trajectories are reserved for threshold and uncertainty calibration. All experiments use random seed 239.

C Evaluation Metrics

Open-loop trajectory metrics. The predicted ego-frame actions are integrated from the final observed pose to obtain a global 6-DoF trajectory. Ego-frame translations are rotated into the global frame using the current yaw, while angular increments are accumulated and wrapped to $[-\pi, \pi]$. Let $\hat{\mathbf{p}}_t$ and $\mathbf{p}_t$ denote the predicted and ground-truth 3-D positions at step t, respectively, and let $H = 8$ denote the prediction horizon. Absolute trajectory error (ATE) and relative pose error (RPE) are computed as

$$\mathrm{ATE} = \sqrt{\frac{1}{H+1}\sum_{t=0}^{H} \|\hat{\mathbf{p}}_t - \mathbf{p}_t\|_2^2}, \tag{9}$$

$$\mathrm{RPE} = \sqrt{\frac{1}{H}\sum_{t=1}^{H} \|(\hat{\mathbf{p}}_t - \hat{\mathbf{p}}_{t-1}) - (\mathbf{p}_t - \mathbf{p}_{t-1})\|_2^2}. \tag{10}$$

Here, RPE measures the error between consecutive translation increments; rotational accuracy is evaluated separately. To account for different trajectory scales, both errors are normalized by the ground-truth path length:

$$L = \sum_{t=1}^{H} \|\mathbf{p}_t - \mathbf{p}_{t-1}\|_2, \qquad \mathrm{rel.ATE} = \frac{\mathrm{ATE}}{L}, \qquad \mathrm{rel.RPE} = \frac{\mathrm{RPE}}{L}. \tag{11}$$

The normalization is performed independently for each trajectory before averaging over the evaluation set. Lower values indicate better trajectory prediction.

Attitude and prediction metrics. Let $\hat{\boldsymbol{\theta}}_t$ and $\boldsymbol{\theta}_t$ denote the predicted and ground-truth roll, pitch, and yaw. Attitude error is computed as the mean absolute wrapped angular difference:

$$E_{\mathrm{att}} = \frac{1}{3(H+1)}\sum_{t=0}^{H}\sum_{k=1}^{3} \left|\mathrm{wrap}\left(\hat{\theta}_{t,k} - \theta_{t,k}\right)\right|. \tag{12}$$

We additionally report the unweighted RMSE over all six action dimensions and the mean absolute error between predicted and target future latent tokens:

$$E_{\mathrm{act}} = \sqrt{\frac{1}{6H}\sum_{t=1}^{H} \|\hat{\mathbf{a}}_t - \mathbf{a}_t\|_2^2}, \qquad E_{\mathrm{latent}} = \frac{1}{N_z}\|\hat{\mathbf{z}} - \mathbf{z}\|_1, \tag{13}$$

where N_z is the number of evaluated latent elements. Action and latent errors are diagnostic metrics, while rel.ATE and rel.RPE are used as the primary trajectory-quality measures.

Closed-loop task metrics. During closed-loop evaluation, only the first action of each predicted plan is executed before obtaining a new observation and replanning. The trajectory metrics above are computed from the complete executed trajectory and its time-aligned reference trajectory without post-hoc spatial or temporal alignment. Task progress is computed by projecting the terminal

position onto the nearest segment of the full reference path and normalizing the resulting arc length by the total reference length. Terminal position and attitude errors are defined as

$$E_{\text{goal}}^{\text{pos}} = \|\mathbf{p}_T - \mathbf{p}_{\text{goal}}\|_2 , \qquad E_{\text{goal}}^{\text{att}} = \frac{1}{3}\sum_{k=1}^{3} |\text{wrap}(\theta_{T,k} - \theta_{\text{goal},k})| . \tag{14}$$

An episode is considered successful only when it completes the prescribed control horizon without collision and terminates within 2 m position error and 0.35 rad attitude error. Success and collision rates are computed as the corresponding fractions over all evaluation episodes.

Computational efficiency. Inference latency is measured with batch size one using synchronized GPU wall-clock time, excluding data loading and image preprocessing. We report mean, median, and 95th-percentile latency. Throughput and the Predictor-step reduction of the adaptive model are calculated as

$$\text{FPS} = \frac{1000}{\bar{t}_{\text{infer}}}, \qquad \text{Reduction} = 1 - \frac{\overline{d}}{H}, \tag{15}$$

where $\bar{t}_{\text{infer}}$ is the mean inference latency in milliseconds and $\overline{d}$ is the mean selected rollout depth. Trajectory metrics are averaged equally across episodes, whereas latency and rollout depth are averaged over all executed control steps.

D DroneNav-6D Dataset Analysis

We analyze the size, trajectory characteristics, six-dimensional motion, and visual appearance of DroneNav-6D. The statistics cover five simulation worlds: CITYBIM, NYC, RuralAustralia, SnappyRoads, and Snowy Mountain. The following analyses use recorded trajectory clips, pose increments, and sampled RGB observations.

D.1 Dataset Size and Composition

We first quantify the amount of recorded flight data and its distribution across environments to characterize the supervision available for learning future prediction.

Table 7 summarizes 3,837 trajectory clips containing 1,592,170 frames, 63.03 hours of flight, and 2,004.11 km of traveled distance. The training and validation sets contain 3,453 and 384 clips, respectively, with distinct clip identifiers and all five worlds represented in both sets. The median recording rate is approximately 7.00 Hz. The clips provide 1,519,267 overlapping windows of 20 consecutive frames, each contained within a single clip. Figure 7 shows the contribution of each world. NYC contributes 603,705 frames, or 37.92% of the corpus, followed by CITYBIM with 373,702 frames, or 23.47%. The other three worlds each contribute between 11.64% and 13.66% of the frames. The overall median clip duration is 57.12 s, with an interquartile range of 49.42–61.43 s.

These statistics establish a corpus of continuous flight sequences with temporal context for multi-step prediction and coverage of all five worlds in the training and validation data.

Table 7: Size of DroneNav-6D. Window counts enumerate overlapping sequences of 20 consecutive frames within individual clips. Flight duration and distance are summed over clips.

Split	Clips	Frames	Hours	Distance (km)	20-frame windows
Training	3,453	1,427,016	56.49	1,795.42	1,361,409
Validation	384	165,154	6.54	208.69	157,858
Total	3,837	1,592,170	63.03	2,004.11	1,519,267

D.2 Six-Dimensional Motion Distributions

We examine the distributions of pose increments to characterize the translation and attitude changes represented in each of the six motion components.

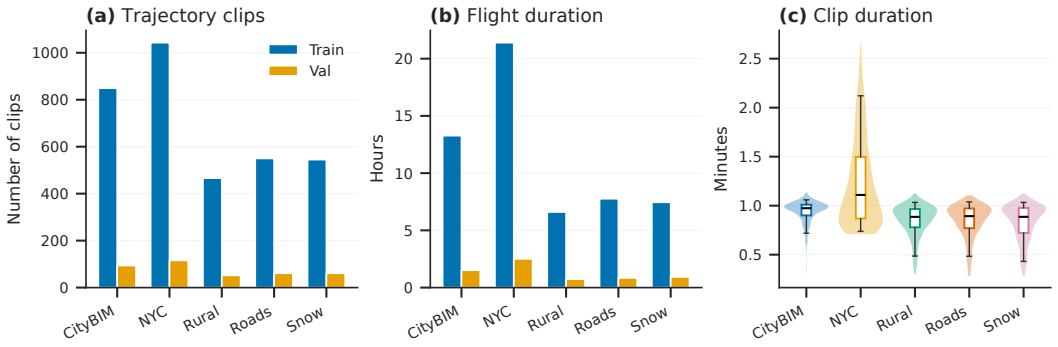


Figure 7: Dataset composition across the five worlds. The panels show (a) training and validation clip counts, (b) recorded flight duration, and (c) the distribution of individual clip durations. CityBIM, Rural, Roads, and Snow denote CITYBIM, RuralAustralia, SnappyRoads, and Snowy Mountain, respectively. These abbreviations are used throughout the dataset figures.

For this analysis, Δx and Δy express translation in horizontal axes aligned with the current yaw, and Δz follows the NED vertical axis. The angular components $\Delta\phi$, $\Delta\theta$, and $\Delta\psi$ are wrapped differences in roll, pitch, and yaw. Figure 8 presents the per-world distributions using clip-balanced samples. Forward translation is concentrated at positive values, while lateral and vertical translation include changes in both directions. All five worlds also exhibit positive and negative roll, pitch, and yaw increments. Table 8 provides the corresponding dataset summary: the median forward increment is 1.138 m per frame, and the 10th–90th percentile intervals are $[-1.395, 1.392]^\circ$ per frame for roll, $[-1.252, 1.376]^\circ$ per frame for pitch, and $[-4.406, 4.376]^\circ$ per frame for yaw. Yaw exhibits a wider central range than roll and pitch.

These distributions show that the recorded motion combines forward flight, lateral and vertical adjustments, and variation in all three attitude components. The resulting pose increments provide supervision for learning future motion across all six degrees of freedom.

Table 8: Distribution of pose increments between consecutive frames. Translation uses yaw-aligned horizontal axes and the NED vertical axis; angular increments are wrapped Euler-angle differences. P10 and P90 denote the 10th and 90th percentiles. Values are rounded to three decimal places.

Component	Unit	P10	Median	P90
Δx	m/frame	0.408	1.138	1.346
Δy	m/frame	−0.943	0.000	0.945
Δz	m/frame	−0.199	0.000	0.193
$\Delta\phi$ (roll)	deg/frame	−1.395	0.000	1.392
$\Delta\theta$ (pitch)	deg/frame	−1.252	0.011	1.376
$\Delta\psi$ (yaw)	deg/frame	−4.406	0.000	4.376

D.3 Trajectory Length and Motion

We compare clip duration, traveled distance, vertical travel, and flight speed to understand how trajectory characteristics vary across environments.

Figure 9 summarizes clip duration, three-dimensional path length, cumulative vertical travel, and the 95th-percentile linear speed within each clip. NYC exhibits wider distributions of duration, path length, and vertical travel. Its median clip duration is 66.57 s, compared with 53.14–58.44 s across the other worlds (Table 9). The path-length medians are approximately 0.5 km, with a longer upper tail in NYC. Cumulative vertical travel also extends to larger values in NYC. Across worlds, the median flight speeds lie in the narrow range of 9.298–9.357 m/s, while typical clip-level 95th-percentile speeds are close to 10 m/s.

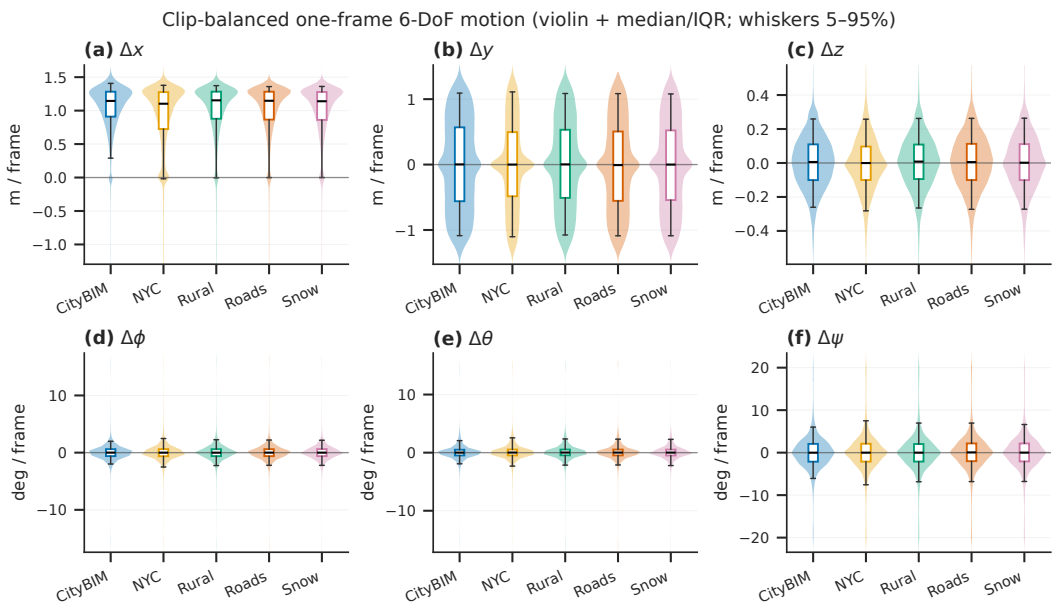


Figure 8: Distributions of six-dimensional pose increments between consecutive frames, shown separately for each world using clip-balanced samples. The top row shows translation in meters per frame, and the bottom row shows roll, pitch, and yaw changes in degrees per frame. Violin plots depict the distributions; boxes indicate the interquartile range, central lines mark the medians, and whiskers span the 5th–95th percentiles.

The dataset therefore combines similar typical flight speeds with variation in sequence duration, traveled distance, and vertical travel. These quantities describe complementary aspects of the motion represented in the collected trajectories.

Table 9: Per-world data and motion statistics. Duration is the median clip duration; speed is the per-world median in the dataset statistics.

World	Clips	Frames	Duration (s)	Speed (m/s)
CITYBIM	943	373,702	58.44	9.357
NYC	1,159	603,705	66.57	9.298
RuralAustralia	518	185,346	53.14	9.319
SnappyRoads	611	217,483	53.58	9.329
Snowy Mountain	606	211,934	53.16	9.325

D.4 Motion over Longer Horizons

We examine how translational and rotational changes grow with the prediction horizon to characterize the motion encountered during multi-step future prediction.

Figure 10 summarizes translational displacement and quaternion-based geodesic rotation over horizons of one to eight frame intervals. The median translational displacement increases approximately linearly from 1.3 m at one interval to 10.6 m at eight intervals. The median rotational displacement grows from approximately 2.7° to 20.7°. The per-world median curves remain close throughout this range. The pooled 10th–90th percentile interval for rotation broadens substantially, reaching approximately 6°–52° at eight intervals. At the median recording rate of 7.00 Hz, the longest horizon corresponds to approximately 1.14 s.

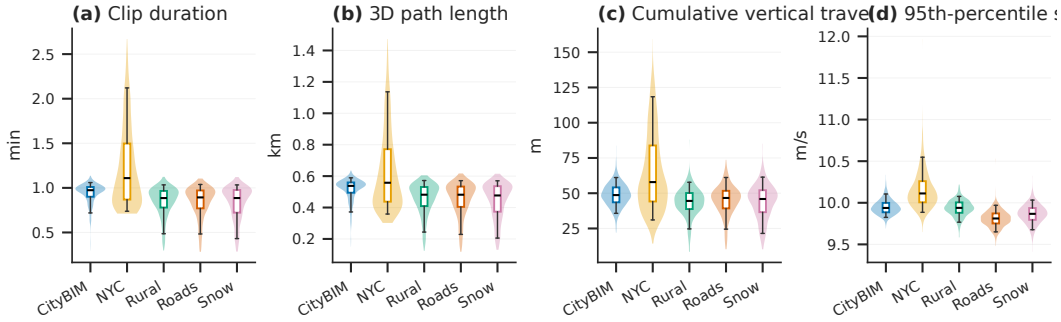


Figure 9: Trajectory characteristics across worlds. Violin and box plots summarize (a) clip duration, (b) three-dimensional path length, (c) cumulative vertical travel, and (d) the 95th-percentile linear speed within each clip.

These observations show that longer prediction horizons cover larger position and attitude changes, together with a broad range of rotational changes across samples. They motivate evaluating multi-step prediction across both the horizon length and the amount of viewpoint change.

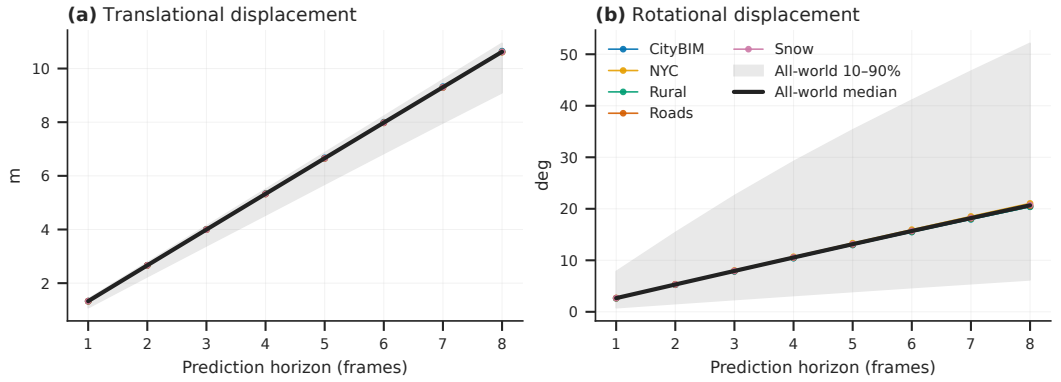


Figure 10: Motion over prediction horizons of one to eight frame intervals. The panels show translational displacement and quaternion-based geodesic rotation. Colored curves are per-world medians; the black curve is the pooled median, and the shaded region spans the pooled 10th–90th percentiles.

D.5 Translation and Rotation

We examine correlations between translation and rotation at the single-frame and clip levels to characterize how these aspects of flight vary together.

Figure 11(a) shows associations between lateral translation and yaw ($\rho = -0.573$), lateral translation and roll ($\rho = -0.413$), and forward translation and pitch ($\rho = 0.297$), using the pose-increment definitions in Section D.2. At the clip level, the 95th-percentile linear and angular speeds have a Spearman correlation of approximately -0.01 (Figure 11(b)). Clips with similar linear speeds span a broad range of angular speeds.

The single-frame statistics identify associations among translational and angular components, and the clip-level results show variation in rotational motion at similar flight speeds. Together, they motivate describing trajectories with both translation and attitude information and reporting angular motion alongside linear speed.

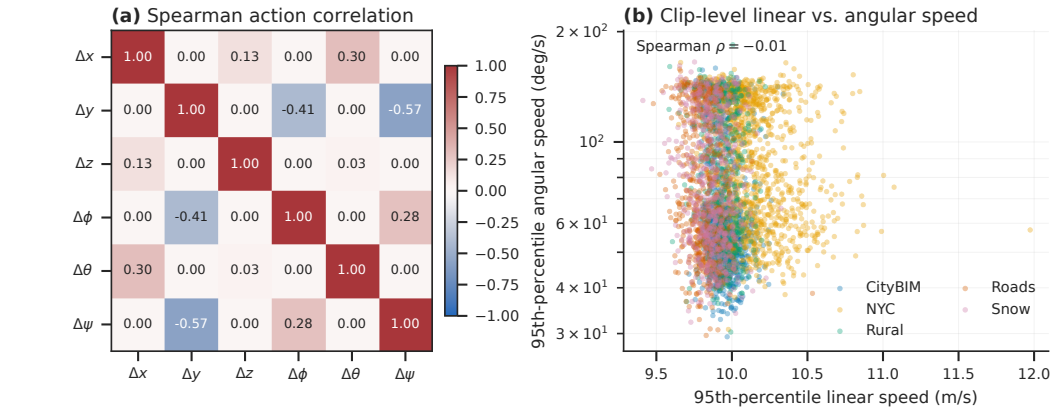


Figure 11: Translation and rotation in the recorded motion. (a) Spearman correlations among the six pose-increment components. (b) Each point shows the 95th-percentile linear and angular speeds of one clip, colored by world. The vertical axis uses a logarithmic scale.

D.6 Visual Appearance across Environments

We examine visual features from sampled RGB observations to characterize appearance variation across worlds and the local feature neighborhoods covered by the training and validation samples.

Figure 12 uses 1,000 images, with 100 training images and 100 validation images sampled from each world. Frozen ImageNet-pretrained AlexNet fc7 features are reduced to 50 dimensions using PCA and then projected into two dimensions using t-SNE. The reported trustworthiness at 10 neighbors is 0.990. When colored by world, samples form groups associated with the individual environments, showing world-specific appearance patterns in this feature representation. Recoloring the same embedding by split shows training and validation samples throughout the local groups.

The visualization provides qualitative evidence of appearance variation across the five worlds and shared local visual coverage between the sampled training and validation images within these environments.

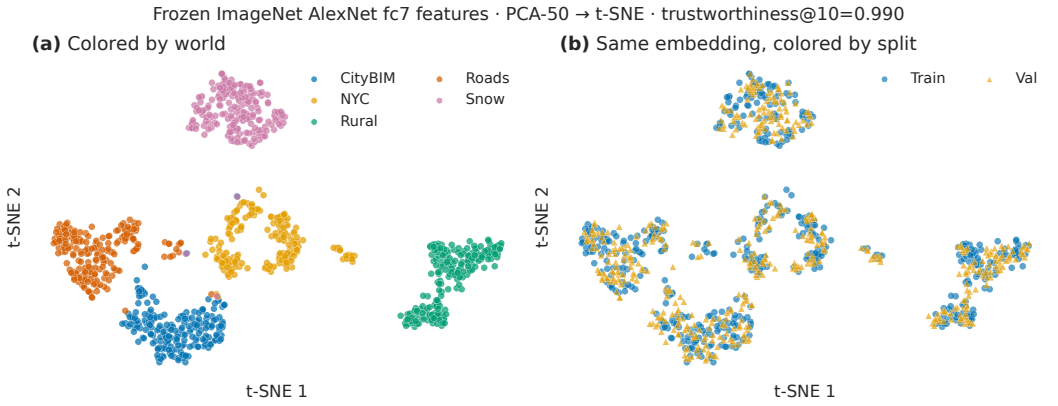


Figure 12: Visual features of 1,000 sampled RGB observations. Each world contributes 100 training and 100 validation images. Frozen ImageNet-pretrained AlexNet fc7 features are projected with PCA to 50 dimensions and then with t-SNE to two dimensions. The same coordinates are colored by (a) world and (b) split. The reported trustworthiness at 10 neighbors is 0.990; the visualization characterizes local feature neighborhoods.

E MORE DATA VISUALIZATION



Figure 13: DroneNav-6D sample 1.

Figures 13 and 14 present two representative goal-conditioned navigation samples from DroneNav-6D, together with the global scene layout, flight trajectory, start observation, intermediate observations, and goal image. In the first example, the drone starts above a vegetation-dominated region and follows a long, curved trajectory across the central park before approaching a building-dominated goal location. The intermediate observations gradually reveal the water body and surrounding skyline, while the goal view differs substantially from the initial observation in both scene appearance and camera viewpoint. In the second example, the drone starts close to urban structures, transitions through an open park region, and eventually reaches a dense building area. Although the trajectory is comparatively smoother, the visual observations still undergo substantial changes in scene composition and viewpoint throughout the flight.

These examples illustrate that image-goal navigation in DroneNav-6D cannot rely on direct visual matching between the current observation and the goal image. Instead, the model needs to capture the underlying motion and scene evolution over extended 6-DoF trajectories in order to predict actions that consistently progress toward the target. More broadly, DroneNav-6D intentionally contains relatively long navigation paths with substantial visual changes between the start and goal observations, making long-horizon navigation from visual observations particularly challenging. This difficulty is also reflected in the relatively limited closed-loop task completion of current models. Nevertheless, we believe that continued advances in world models, especially in long-horizon prediction and action-conditioned imagination, will progressively enable more reliable long-range navigation from visual observations alone.

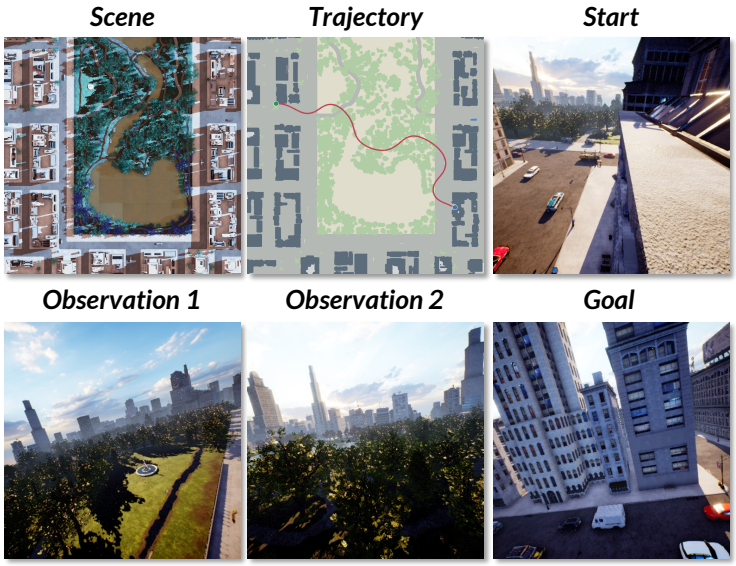


Figure 14: DroneNav-6D sample 2.

F Additional Experimental Results

Closed-loop Results on Easy Routes. The challenging routes in our original closed-loop benchmark require long-horizon visual localization and repeated replanning, under which none of the evaluated methods successfully completes an episode. To further examine whether the models can nevertheless support effective closed-loop navigation under a more tractable setting, we construct an additional easy-route test set containing 50 trajectories. These routes consist primarily of straight flight segments and mild turns, reducing the difficulty of long-horizon visual localization while retaining the same closed-loop perception–prediction–action evaluation protocol.As shown in Table 10, DroneWAM achieves an 88% success rate, outperforming NWM and FastWAM by 14 and 16 percentage points, respectively. It also substantially reduces trajectory error, achieving a rel.ATE of 0.15 and a rel.RPE of 0.008, compared with 0.31/0.012 for NWM and 0.38/0.014 for FastWAM. Meanwhile, DroneWAM reaches a task progress of 0.95, indicating that its predicted actions remain well aligned with the reference route throughout closed-loop execution. Importantly, these gains are obtained with an inference latency of 460 ms, lower than FastWAM's 700 ms and substantially lower than NWM. These results verify that DroneWAM can support successful closed-loop visual navigation when route complexity is moderate. Together with the results on the full benchmark, they suggest that the remaining failures are primarily associated with challenging long-horizon visual localization and accumulated control errors, rather than an inability of the model to operate in a closed-loop manner.

Table 10: Global visual closed-loop results on easy routes.

Method	Success ↑	rel.ATE ↓	rel.RPE ↓	Task Progress ↑	Latency (ms) ↓
NWM	74%	0.31	0.012	0.79	1.3×10^5
FastWAM	72%	0.38	0.014	0.76	700ms
DroneWAM	88%	0.15	0.008	0.95	460ms

Quantitative Case Studies of Different Rollout Depths. To better understand how rollout depth affects individual predictions, we examine four representative held-out examples. For each example, we report the prediction errors at depth 1, the depth selected by the Gate, and depth 8. As shown in Table 11, the effect of additional rollout varies across samples. In Case 1, increasing the rollout depth substantially improves the prediction, reducing rel.ATE from 0.1879 at depth 1 to 0.1083 at the Gate-selected depth 4, with only a small further improvement at depth 8. Case 2 shows the opposite trend: the prediction is already accurate at depth 1, while deeper rollout slightly increases both rel.ATE and rel.RPE. In Case 3, the Gate selects depth 2, where the lowest error is obtained; continuing to depth 8 instead increases rel.ATE from 0.0423 to 0.0514. Case 4 illustrates a failure case, where the Gate stops at depth 1 although deeper rollout would lead to a substantially lower error. These examples further show that the preferred rollout depth is sample-dependent. Some observations benefit from additional prediction, while others are already sufficiently resolved after only a few steps. They also reveal a remaining limitation of the learned Gate: inaccurate early stopping can occasionally prevent the model from benefiting from deeper rollout.

Table 11: Case studies at different rollout depths. Each row evaluates the same sample at depth 1, the Gate-selected depth, and depth 8.

Case	Gate depth	D1 rel.ATE	Gate rel.ATE	D8 rel.ATE	D1 rel.RPE	Gate rel.RPE	D8 rel.RPE
Case 1	4	0.1879	0.1083	**0.1064**	0.0627	0.0366	**0.0322**
Case 2	1	**0.0142**	**0.0142**	0.0192	**0.0064**	**0.0064**	0.0073
Case 3	2	0.0543	**0.0423**	0.0514	0.0164	**0.0127**	0.0150
Case 4	1	0.2160	0.2160	**0.1285**	0.0924	0.0924	**0.0668**

Does Adaptive Rollout Transfer Beyond Aerial Motion? The main paper reports only normalized RMSE, gripper accuracy, and latency on LIBERO. To verify that the conclusion is consistent across action metrics, we evaluate normalized RMSE and L1 over all seven action dimensions, raw-unit end-effector RMSE and L1 over the first six dimensions, gripper-state accuracy, mean rollout

depth, and synchronized batch-one latency. The episode-level split contains 173 held-out trajectories, and every method predicts the same 32-step action chunk from the same deterministic center window. As shown in Table 12, adaptive rollout consistently improves the accuracy–efficiency tradeoff. Compared with Base WAM, our method reduces normalized RMSE from 0.3636 to 0.2652 and latency from 957.1 ms to 411.9 ms, corresponding to reductions of 27.1% and 57.0%, respectively. Compared with Fixed D8, it reduces latency by 53.8%, while further improving normalized RMSE from 0.2685 to 0.2652 and gripper accuracy from 89.83% to 90.55%, an increase of 0.72 percentage points. The same trend is observed for normalized L1 and both end-effector metrics. Importantly, the adaptive model achieves the best action-prediction accuracy across all reported metrics with an average rollout depth of only 3.56. It also outperforms Fixed D4 in both prediction quality and latency, despite using fewer rollout steps on average. These results indicate that the gain does not arise from simply truncating imagination, but from allocating predictive computation according to the input. Overall, adaptive rollout transfers effectively beyond aerial navigation to robotic manipulation, reducing computation without sacrificing action-prediction quality.

Table 12: Full open-loop action-prediction results.

Method	Mean depth	Norm. RMSE ↓	Norm. L1 ↓	EEF RMSE ↓	EEF L1 ↓	Gripper acc. ↑	Latency (ms) ↓
Base WAM	–	0.3636	0.2466	0.2647	0.1631	85.42%	957.1
Fixed D1	1.00	0.2735	0.1874	0.1932	0.1232	89.79%	**129.7**
Fixed D2	2.00	0.2680	0.1844	0.1906	0.1218	90.21%	237.7
Fixed D4	4.00	0.2667	0.1839	0.1890	0.1212	90.35%	451.3
Fixed D8	8.00	0.2685	0.1848	0.1905	0.1223	89.83%	890.8
Ours	3.56	**0.2652**	**0.1825**	**0.1881**	**0.1206**	**90.55%**	411.9

Does the gain hold across all LIBERO suites? We next disaggregate normalized action RMSE by suite to test whether the aggregate result is driven by one task family. Table 13 shows that Adaptive WAM improves over Fixed D8 on Spatial, Object, Goal, and Long, with absolute reductions of 0.0046, 0.0026, 0.0037, and 0.0021, respectively. The selected depth varies from 3.09 on Goal to 4.02 on Object, demonstrating that the Gate changes its compute allocation across task distributions. Goal and Long remain substantially harder in absolute error, but the adaptive advantage is consistent across all four suites. This result supports method-level transfer rather than a gain restricted to a single manipulation category.

Table 13: Normalized action RMSE by LIBERO suite.

Suite	Episodes	Base WAM ↓	Fixed D8 ↓	Ours ↓	Mean depth
Spatial	44	0.3479	0.2321	**0.2275**	3.43
Object	46	0.2609	0.1857	**0.1831**	4.02
Goal	44	0.4461	0.3478	**0.3441**	3.09
Long	39	0.4094	0.3177	**0.3156**	3.69

How to Improve Closed-Loop Long-Horizon Manipulation? Our previous LIBERO experiments mainly evaluate open-loop action prediction, where lower action error does not necessarily translate into successful closed-loop execution. In long-horizon manipulation, small prediction errors can accumulate over repeated execution and replanning, while visually similar observations may correspond to different tasks depending on the language instruction. This makes explicit task conditioning particularly important for maintaining goal-consistent behavior over long horizons.

To improve task disambiguation, we inject the language instruction into both latent imagination and action planning. Specifically, a task-goal encoder maps the instruction to a compact task embedding, which is supplied to a task-conditioned future adapter to modulate the autoregressively predicted latent tokens. The same task embedding is also provided to the iterative Planner, allowing action decoding to depend jointly on the observed visual context, imagined future, robot state, executed-action history, and task specification. We retain the causal closed-loop interface. The improved policy uses four causal dual-view RGB observations, a fixed rollout depth of $D = 2$, and a 32-step predicted action chunk, of which the first 16 controls are executed before replanning from fresh observations. The visual encoder, TokenAE resampler, and pretrained WAM backbone remain frozen,

while the task-goal encoder, task-conditioned future adapter, Plan Adapter, and action generator are optimized on the LIBERO training demonstrations.

As shown in Table 14, explicit task conditioning substantially improves closed-loop manipulation across all four LIBERO suites. The average success rate increases from 45.23% to 59.94%, corresponding to an absolute gain of 14.71 percentage points. The largest improvement is observed on Object, where success rises from 64.32% to 90.22%, followed by Goal with a gain of 15.95 points and Spatial with a gain of 10.63 points. These improvements are consistent with the role of language conditioning in disambiguating task intent and keeping both imagined futures and action plans aligned with the specified goal.

The Long suite remains the most challenging setting, although its success rate improves from 10.08% to 16.44%. This suggests that task conditioning alleviates goal ambiguity but does not fully resolve the accumulated prediction and control errors that arise over extended closed-loop horizons. More broadly, the results indicate that accurate open-loop action prediction alone is insufficient for long-horizon manipulation: effective closed-loop behavior also requires the world model to preserve task-relevant information throughout imagination and planning. We expect further improvements in long-horizon world modeling, action-conditioned prediction, and closed-loop replanning to progressively narrow this gap.

Table 14: Closed-loop success rates (%) on LIBERO.

Method	Spatial	Object	Goal	Long	Avg.
DroneWAM	52.26	64.32	54.27	10.08	45.23
Improved-DroneWAM	**62.89**	**90.22**	**70.22**	**16.44**	**59.94**

G ALGORITHM

Algorithm 1 Training DroneWAM

Require: Training set $\mathcal{D}_{\mathrm{tr}}$, calibration set $\mathcal{D}_{\mathrm{cal}}$, frozen encoder E, maximum rollout depth $K_{\max}$, action horizon H

Ensure: Resampler R_ρ, Predictor F_θ, Planner P_ψ, Gate p_ω, and thresholds $\{\eta_k\}_{k=K_{\min}}^{K_{\max}-1}$

Stage I: Resampler pretraining

for each minibatch of video clips from $\mathcal{D}_{\mathrm{tr}}$ **do**
  Extract dense visual tokens $\mathbf{x}_{1:T} \leftarrow E(\mathbf{o}_{1:T})$
  Compress tokens $\mathbf{z}_{1:T} \leftarrow R_\rho(\mathbf{x}_{1:T})$
  Reconstruct $\widetilde{\mathbf{x}}_{1:T} \leftarrow \mathrm{Decoder}(\mathbf{z}_{1:T})$
  Update R_ρ and Decoder using feature-reconstruction and temporal-consistency losses
end for
Discard Decoder and freeze E and R_ρ

Stage II: Predictor–Planner training

for each $(\mathbf{o}_{\le t}, \mathbf{s}_{\le t}, \mathbf{a}_{\le t}, g)$ from $\mathcal{D}_{\mathrm{tr}}$ **do**
  $\mathbf{z}_{\le t} \leftarrow R_\rho(E(\mathbf{o}_{\le t}))$ and $\mathbf{z}_g \leftarrow R_\rho(E(g))$
  Form the causal context $C_t \leftarrow \{\mathbf{z}_{\le t}, \mathbf{s}_{\le t}, \mathbf{a}_{\le t}\}$
  for $k = 1, \ldots, K_{\max}$ **do**
    $\widehat{\mathbf{z}}_{t+k} \leftarrow F_\theta(C_t, \mathbf{z}_g, \widehat{\mathbf{z}}_{t+1:t+k-1})$
    $\widehat{\mathbf{A}}_t^{(k)} \leftarrow P_\psi(\widehat{\mathbf{z}}_{t+1:t+k}, \mathbf{z}_g)$
    Integrate $\widehat{\mathbf{A}}_t^{(k)}$ from the current pose to obtain its predicted trajectory
  end for
  Compute the discounted latent loss $\mathcal{L}_{\mathrm{WM}}$
  Compute $\mathcal{L}_{\mathrm{PLA}}$ by averaging normalized rATE and rRPE over all depths
  Update θ and ψ using $\mathcal{L}_{\mathrm{WAM}} = \lambda_{\mathrm{WM}}\mathcal{L}_{\mathrm{WM}} + \lambda_{\mathrm{PLA}}\mathcal{L}_{\mathrm{PLA}}$
end for
Freeze E, R_ρ, F_θ, and P_ψ

Stage III: Gate preference training

for each sample i in $\mathcal{D}_{\mathrm{tr}}$ **do**
  for $k = 1, \ldots, K_{\max}$ **do**
    Generate $\widehat{\mathbf{z}}_{t+1:t+k}$, $\widehat{\mathbf{A}}_t^{(k)}$, and the causal Gate input $\mathbf{u}_{i,k}$
    $C_{i,k} \leftarrow \mathrm{rATE}_{i,k}/\epsilon_A + \mathrm{rRPE}_{i,k}/\epsilon_R + \lambda_c k/K_{\max}$
  end for
  $k_i^+ \leftarrow \arg\min_k C_{i,k}$
  Add $(\mathbf{u}_{i,k_i^+}, \mathbf{u}_{i,k^-})$ to $\mathcal{D}_{\mathrm{pref}}$ for every $k^- \neq k_i^+$
end for
Initialize and freeze the reference Gate p_{ref} from p_ω
for each preference minibatch $(\mathbf{u}^+, \mathbf{u}^-)$ from $\mathcal{D}_{\mathrm{pref}}$ **do**
  Compute $r_\omega(\mathbf{u}^+)$ and $r_\omega(\mathbf{u}^-)$ relative to p_{ref}
  Update only ω by minimizing $\mathcal{L}_{\mathrm{DPO}}$
end for
Calibrate the depth-specific thresholds $\{\eta_k\}$ on $\mathcal{D}_{\mathrm{cal}}$

Algorithm 2 Adaptive DroneWAM Inference

Require: Observations $\mathbf{o}_{\leq t}$, motion states $\mathbf{s}_{\leq t}$, executed actions $\mathbf{a}_{\leq t}$, goal image g, thresholds $\{\eta_k\}$, and depths $K_{\min}, K_{\max}$

Ensure: Executed action $\widehat{\mathbf{a}}_{t+1|t}^{(\kappa_t)}$, stopping depth τ_t, and selected-plan depth κ_t

$\mathbf{z}_{\leq t} \leftarrow R_\rho(E(\mathbf{o}_{\leq t}))$ and $\mathbf{z}_g \leftarrow R_\rho(E(g))$
$C_t \leftarrow \{\mathbf{z}_{\leq t}, \mathbf{s}_{\leq t}, \mathbf{a}_{\leq t}\}$
Initialize $\tau_t \leftarrow K_{\max}$ and $\kappa_t \leftarrow 1$
for $k = 1, \ldots, K_{\max}$ **do**
    $\widehat{\mathbf{z}}_{t+k} \leftarrow F_\theta(C_t, \mathbf{z}_g, \widehat{\mathbf{z}}_{t+1:t+k-1})$
    $\widehat{\mathbf{A}}_t^{(k)} \leftarrow P_\psi(\widehat{\mathbf{z}}_{t+1:t+k}, \mathbf{z}_g)$
    Construct the causal Gate input $\mathbf{u}_{t,k}$ from C_t, $\widehat{\mathbf{z}}_{t+1:t+k}$, and $\widehat{\mathbf{A}}_t^{(1:k)}$
    $q_{t,k} \leftarrow \sigma(\beta r_\omega(\mathbf{u}_{t,k}))$
    if $k = 1$ or $q_{t,k} > q_{t,\kappa_t}$ **then**
        $\kappa_t \leftarrow k$
    end if
    if $k \geq K_{\min}$ and $k < K_{\max}$ and $q_{t,k} \geq \eta_k$ **then**
        $\tau_t \leftarrow k$; **break**
    end if
end for
Execute only $\widehat{\mathbf{a}}_{t+1|t}^{(\kappa_t)}$, the first action of $\widehat{\mathbf{A}}_t^{(\kappa_t)}$
Receive the next observation and repeat in a receding-horizon manner