FaB-Fly: Foresight and Backtracking for Aerial Vision-and-Language Navigation

Anonymous authors
Paper under double-blind review

Abstract

In aerial vision-and-language navigation, an unmanned aerial vehicle (UAV) follows natural-language instructions that specify landmarks and how to move relative to them. Recent memory-based hierarchical approaches retrieve candidate landmark locations from memory and use a planner to select successive targets. However, several locations may match the same landmark description, so the planner must assess whether flying to each candidate would satisfy the instruction. If the UAV moves on after arrival without checking whether its flight followed the instruction, an incorrect choice can lead to further navigation errors. We propose FaB-Fly, a hierarchical navigation framework that uses *Foresight* to select targets and *Backtracking* to recover from incorrect target choices. Foresight uses a video world model to generate a flight video for each candidate, based on current observations and a candidate-specific movement instruction. A multimodal large language model (MLLM) scores how well each predicted video matches the corresponding part of the navigation instruction to guide target selection. After execution, Backtracking evaluates the observed flight against the same instruction segment. If the score is low, the UAV returns to the departure point and retries with another candidate. Experiments on AerialVLN-S and AerialVLN-Fine show that FaB-Fly outperforms the strongest baselines in success rate. Further analysis shows that FaB-Fly achieves a success rate close to that of exhaustively visiting every candidate while requiring about half the computation time.

1 Introduction

Aerial Vision-and-Language Navigation (Aerial VLN) requires an unmanned aerial vehicle (UAV) to follow natural-language instructions through a three-dimensional environment (Liu et al., 2023; Wang et al., 2025). These instructions specify both the landmarks to reach and how to move relative to them, as in "Fly over the river toward a white building, and then turn left to approach the gas station." Following such instructions requires the UAV not only to reach the described landmarks (e.g., the white building and the gas station) but also to move as described along the way.

Recent hierarchical approaches to Aerial VLN separate subgoal planning (i.e., where to fly) from low-level action generation (i.e., how to fly) (Zhang et al., 2025a; Ning et al., 2026; Shao et al., 2026). To plan beyond the current view, some of these approaches store observed landmarks and their locations in a landmark memory and retrieve locations that match the landmark descriptions in the instruction as candidate targets (Zhang et al., 2025a; Ning et al., 2026). However, this design faces two challenges (Figure 1). First, simply choosing a location that matches a landmark description does not ensure that the flight to it satisfies the instructed movement: in the example above, the environment may contain several white buildings, but only some of them are reached by flying over the river. Second, an incorrect choice propagates to subsequent navigation: after reaching the wrong white building, turning left may lead the UAV to a different gas station.

The first challenge is to assess candidate flights before selecting a target (Figure 1(a)). Whether a flight satisfies the instructed movement depends on the path the UAV would take and what it would pass along the way, which cannot be observed when the target is selected. In the example, the planner must anticipate whether flying to each white building would take the UAV over the river. Existing approaches make this judgment from spatial representations that encode where landmarks lie relative to one another, such as top-down semantic maps (Gao et al., 2025), topological memory (Zhang

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

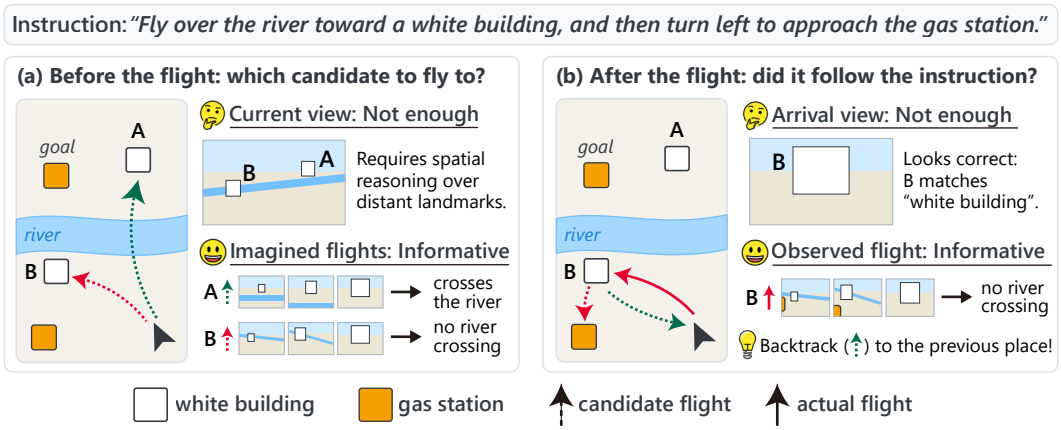

Figure 1: **Two challenges in hierarchical Aerial VLN.** (a) Both white buildings match the landmark description; the current view alone makes it hard to tell which flight crosses the river, whereas imagined flights make it evident. (b) On arrival, the wrong building B looks correct, and continuing from B leads to the wrong gas station; the observed flight reveals the error, allowing the UAV to backtrack.

et al., 2025a), and scene graphs (Xu et al., 2026). More recently, LookasideVLN (Ning et al., 2026) uses a multimodal large language model (MLLM) as a planner to select target locations, focusing on whether the direction toward each candidate is consistent with the instruction. However, such spatial relations indicate where each candidate lies, but not what the UAV would pass on its way there. This motivates explicitly predicting the flight to each candidate and using the prediction to select the target.

The second challenge is to assess the completed flight and determine whether recovery is needed (Figure 1(b)). Flight prediction can be inaccurate for distant or unobserved regions. As a result, the UAV may reach the selected location without following the current instruction segment, although its predicted flight appeared to follow it. Most existing hierarchical approaches move on to the next target once the UAV arrives (Zhang et al., 2025a; Ning et al., 2026). FineCog-Nav (Shao et al., 2026) additionally checks subgoal completion before moving on, based on the observation on arrival. However, such arrival-based checks alone may not reveal a wrong choice: in the example, a white building on the near side of the river matches the expected scene on arrival, even though the UAV never crossed the river to reach it. This motivates assessing the observed flight against the current instruction segment to decide whether to accept the selected target or retry the segment with another candidate.

To address these challenges, we propose FaB-Fly, a hierarchical navigation framework that combines *Foresight* for target selection with *Backtracking* for recovery (Figure 2). The key idea is to assess flights against the current instruction segment both before and after execution: predicted flights guide target selection, and the observed flight determines whether to retry the segment. Foresight uses a video world model to predict flights to candidate locations retrieved from a landmark memory, as in prior hierarchical approaches. The predictions are conditioned on current observations and candidate-specific movement instructions. An MLLM judge scores the predicted videos against the current instruction segment to guide target selection, and a separate low-level controller flies toward the selected location. After the first flight for a segment, Backtracking scores the observed flight against the same segment. If the score falls below a threshold, the UAV returns to the departure point and tries one alternative candidate for the current instruction segment before advancing to the next segment.

We evaluate FaB-Fly on AerialVLN-S (Liu et al., 2023) and AerialVLN-Fine (Shao et al., 2026), measuring navigation success and adherence to the instructed route. Results show that FaB-Fly outperforms the strongest baselines in success rate on both benchmarks. Component ablations show that Foresight and Backtracking work complementarily in reducing navigation errors. Further anal-

yses examine the effects of flight videos and the judge MLLM, and show that FaB-Fly offers a better trade-off between cost and performance than exhaustively visiting every candidate.

Our contributions are summarized as follows:

- We introduce FaB-Fly, a zero-shot hierarchical framework for Aerial VLN that assesses flights against the current instruction segment both before and after execution.
- We propose *Foresight*, which predicts the flight to each candidate location with a video world model conditioned on a candidate-specific movement instruction, and selects the target whose predicted flight best matches the instruction segment.
- We propose *Backtracking*, which evaluates the observed flight against the instruction segment and, when the flight is inconsistent, returns the UAV to its departure point to retry with an alternative candidate.
- Experiments on AerialVLN-S and AerialVLN-Fine show that FaB-Fly outperforms the strongest baselines in success rate, that Foresight and Backtracking work complementarily, and that FaB-Fly offers a better cost–performance trade-off than exhaustively visiting every candidate.

2 Related Work

Aerial Vision-and-Language Navigation. Aerial VLN benchmarks cover instruction following in three-dimensional environments (Liu et al., 2023; Wang et al., 2025; Gao et al., 2026). CityNav studies navigation to destinations described through landmarks and geographic references (Lee et al., 2025). Landmark-based decomposition was explored in ground navigation (Shah et al., 2023), while aerial approaches use semantic maps (Gao et al., 2025) and hierarchical planning (Zhang et al., 2025a; Ning et al., 2026; Shao et al., 2026) to connect instruction-level goals to executable movements. Within hierarchical systems, memories store historical trajectories (Zhang et al., 2025a), associate landmark descriptions with candidate locations (Ning et al., 2026), or organize observations around subgoals and instruction segments (Shao et al., 2026). FaB-Fly retrieves candidates from a landmark memory constructed through a prior visual survey and studies how evaluating their associated flights supports target selection and recovery.

Visual Lookahead for Navigation. Visual lookahead lets navigation agents evaluate possible movements by predicting their visual consequences, using learned dynamics, panoramic observations, semantic features, or latent states (Wang et al., 2018; Koh et al., 2021; Wang et al., 2023; 2024; Yao et al., 2025; Huang et al., 2025; Wang et al., 2026b), or future spatial representations in aerial VLN (Liu et al., 2026b). Recent work uses generative world models in three main ways. Some couple prediction with action generation, jointly predicting future latent observations and actions for UAVs (Zhao et al., 2026; Zheng et al., 2026) and in other navigation settings (Dong et al., 2026; Mei et al., 2026; Azuma et al., 2026). Others treat the predicted video as a plan to execute: ImagineUAV extracts poses from instruction-conditioned videos for kinodynamic planning (Liu et al., 2026a), and NavDreamer samples several videos for the same instruction, scores them with a VLM, and decodes waypoints from the best one (Huang et al., 2026). Still others rank candidate actions by their predicted views, as in navigation world models for image-goal navigation (Bar et al., 2025) and ANWM, which is trained on UAV trajectories (Zhang et al., 2025b). In contrast, FaB-Fly uses an off-the-shelf video model, without aerial-specific training, only to compare locations retrieved for the same landmark description, while a separate controller executes the flight. Moreover, the video model is conditioned on a candidate-specific movement instruction rather than on the instruction being evaluated, so that each prediction reflects where the candidate lies instead of depicting the instructed motion for every candidate.

Progress Monitoring and Navigation Recovery. In ground VLN, progress estimation guides search and backtracking to previously visited viewpoints (Ma et al., 2019a;b; Ke et al., 2019), and MLLM-based agents decide when to backtrack from their navigation history (Shi et al., 2025). Other agents assess completion using observation–action histories, expert discussions, or explicit object, location, and direction constraints (Zhou et al., 2024; Long et al., 2024; Chen et al., 2025). In aerial VLN, FineCog-Nav judges subgoal completion by combining observations, subgoal memory, and

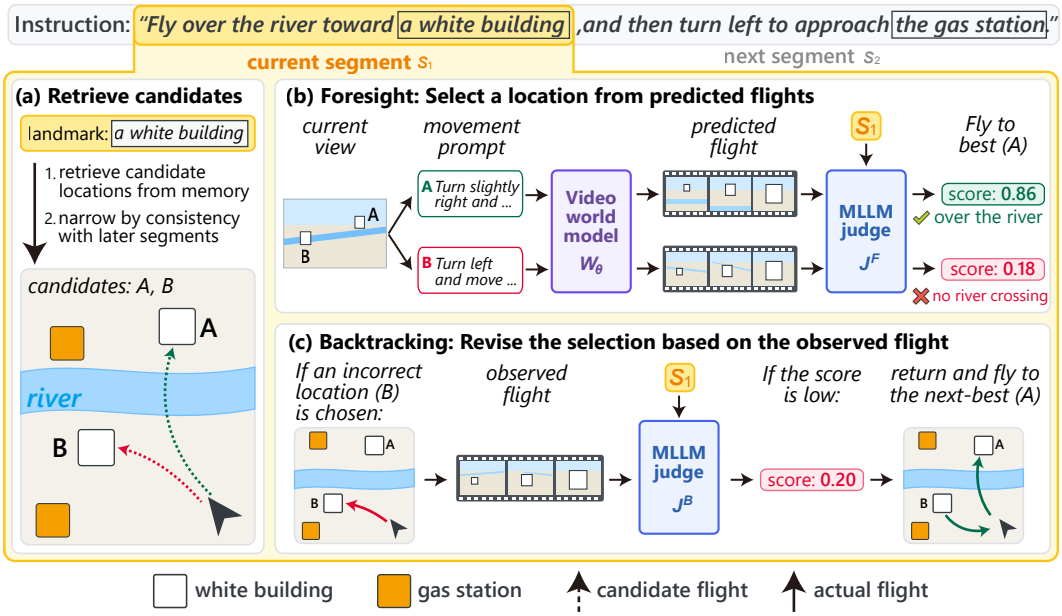

Figure 2: **Overview of FaB-Fly.** (a) FaB-Fly first retrieves candidate locations for the landmark in the current segment from the landmark memory. (b) *Foresight* predicts the flight to each candidate with a video world model and selects the one that an MLLM judge scores highest. (c) *Backtracking* evaluates the observed flight against the same segment and, if the match is poor, returns and retries with an alternative candidate.

imagined scene descriptions (Shao et al., 2026), while other approaches revise action proposals before execution (Li et al., 2026) or replay failures during reinforcement learning (Wang et al., 2026a). Recovery after execution is more demanding for UAVs, since an error may become evident only after a long flight in continuous three-dimensional space, and each return is costly. FaB-Fly addresses this by evaluating the observed flight against the current instruction segment to decide whether to return and retry with another retrieved candidate.

3 Method

We present **FaB-Fly**, a hierarchical Aerial VLN framework that evaluates candidate flights against the current instruction segment both before and after execution (Figure 2). For each instruction segment, a high-level planner selects a location to visit among candidates in the landmark memory, and a low-level controller flies the UAV to the selected location. *Foresight* predicts the flight to each candidate location with a video world model and selects the candidate whose predicted flight best matches the instruction segment. After execution, *Backtracking* evaluates the observed flight against the same segment and, if the match is poor, returns the UAV to the previous point and retries with an alternative candidate. Algorithm 1 summarizes the overall procedure.

3.1 Problem Setup

We consider Aerial VLN, in which a UAV follows a natural-language instruction x along the described route and stops at the described destination. At each step during navigation, the UAV observes its current visual view o and pose q and selects actions based on the given instruction x and its navigation history.

Following prior approaches (Zhang et al., 2025a; Ning et al., 2026), we assume access to a prior visual survey of the environment, from which an instruction-independent landmark memory $\mathcal{M} =$

Algorithm 1 Navigation with FaB-Fly.

1: $(s_i, m_i, \mathcal{C}_i)_{i=1}^{K} \leftarrow$ GROUNDINSTRUCTION$(x, \mathcal{M})$
2: **for** $i = 1, \ldots, K$ **do**
3: $\quad \widetilde{\mathcal{C}}_i \leftarrow$ NARROWCANDIDATES$((m_j, \mathcal{C}_j)_{j=i}^{K})$
4: $\quad (g^{\mathrm{F}}, \widehat{z}) \leftarrow$ FORESIGHT$(s_i, \widetilde{\mathcal{C}}_i)$
5: $\quad V^{\mathrm{obs}} \leftarrow$ FLYTO(g^{F})
6: $\quad$ **if** BACKTRACKING(s_i, V^{obs}) and an alternative exists **then**
7: $\quad\quad$ RETURNTOPREVIOUS()
8: $\quad\quad g \leftarrow \arg\max_{g \in \widetilde{\mathcal{C}}_i \setminus \{g^{\mathrm{F}}\}} \widehat{z}(g)$
9: $\quad\quad$ FLYTO(g)
10: $\quad$ **end if**
11: **end for**

$\{(d_n, g_n)\}_{n=1}^{N}$ is constructed, where d_n and g_n denote the description and estimated location of landmark n (Appendix A.1).

3.2 NAVIGATION FRAMEWORK

The instruction x is parsed by an MLLM into K ordered segments $s_1, \ldots, s_K$, each associated with a landmark phrase ℓ_i and, when present, a motion phrase m_i describing how to reach it. For each segment s_i, memory entries whose landmark descriptions match ℓ_i are retrieved, and their locations form the candidate location set $\mathcal{C}_i$ (GROUNDINSTRUCTION in Alg. 1, line 1). Before planning each flight, a lightweight geometric filter narrows $\mathcal{C}_i$ by considering how candidate routes through the remaining segments agree with their instructed motions, yielding $\widetilde{\mathcal{C}}_i$. The high-level planner selects a target from $\widetilde{\mathcal{C}}_i$, and the low-level controller moves the UAV to it; this process repeats for each instruction segment. Please see Appendix A for the detailed descriptions of the components.

Candidate narrowing. Rather than evaluating each location in $\mathcal{C}_i$ in isolation, we first use a lightweight geometric heuristic to check whether the rest of the instruction can be followed naturally after visiting it. For each $g \in \mathcal{C}_i$, we form possible routes that continue through the candidate sets $\mathcal{C}_{i+1}, \ldots, \mathcal{C}_K$ of the later landmarks, and rank them using a geometric cost based on travel distance and the turning directions given by the motion phrases $m_i, \ldots, m_K$. The locations in $\mathcal{C}_i$ that start the highest-ranked routes form $\widetilde{\mathcal{C}}_i$, containing at most eight candidates for the high-level planner. Details of the route scoring and search are provided in Appendix A.3.

Low-level control. Given a target location selected by the high-level planner, an MLLM-based low-level controller iteratively generates navigation actions using the current observation, the instruction segment s_i, and the target's direction and distance relative to the UAV. The observations collected during the flight form V^{obs}, which Backtracking (Section 3.4) uses to verify whether the executed flight followed s_i. After the flight for the last segment s_K, the UAV stops.

3.3 FORESIGHT: SELECTING LOCATIONS FROM PREDICTED FLIGHTS

Foresight selects the next target from the narrowed candidate locations $\widetilde{\mathcal{C}}_i$ by evaluating the flight predicted for each candidate against the current instruction segment s_i. The geometric narrowing above checks coarse route consistency from landmark locations alone, but does not relate the route to the surrounding scene, such as whether the flight toward a candidate crosses the river. For each $g \in \widetilde{\mathcal{C}}_i$, we therefore use a video world model to predict the flight from the current observation to g, and an MLLM judge scores how well the predicted flight matches s_i.

Candidate flight prediction. For each candidate location $g \in \widetilde{\mathcal{C}}_i$, we construct a candidate-specific movement instruction $u(g)$ from *i*) its direction and distance relative to the current UAV pose and *ii*) its landmark description $d(g)$. The result is a short instruction such as "slightly turn right, and fly forward about 80 m toward the blue stockroom."

Given the current observation o and $u(g)$, a video world model W_θ predicts the corresponding flight:

$$\widehat{V}(g) \sim W_\theta(\cdot \mid o, u(g)). \tag{1}$$

The resulting video represents what the UAV is expected to observe while flying to g.

Predicted flight evaluation. A foresight judge J^{F}, implemented by an MLLM, scores how well each predicted flight matches the instruction segment s_i:

$$\widehat{z}(g) = J^{\mathrm{F}}\big(s_i, \widehat{V}(g)\big) \in [0, 1], \tag{2}$$

where a higher value indicates that the predicted flight better matches s_i.

Note that the world model is conditioned on the candidate-specific movement instruction $u(g)$ rather than on s_i itself: $u(g)$ specifies where to fly but does not reveal the instructed motion or spatial relations being evaluated. If the world model were conditioned on s_i, it could depict the instructed motion (e.g., crossing the river) for every candidate regardless of where the candidate lies, making the predictions uninformative for comparing candidates. Conditioning on $u(g)$ instead makes each predicted flight depend on where the candidate lies, so that $\widehat{z}(g)$ reflects whether flying to that candidate agrees with s_i.

Next location selection. We select $g^{\mathrm{F}} = \arg\max_{g \in \widetilde{\mathcal{C}}_i} \widehat{z}(g)$ as the next target and retain the candidate scores for possible reuse by Backtracking (Section 3.4). The low-level controller then flies the UAV to g^{F}.

3.4 Backtracking: Revising Selections from Observed Flights

Foresight uses predicted flights to improve target selection (Section 3.3), but its decision relies on predicted rather than executed flights, which can still lead it to select an incorrect target. Backtracking therefore uses the observations V^{obs} obtained after actually flying to the selected target to verify whether the executed flight followed the instruction segment s_i. If the observed flight is inconsistent with s_i, the UAV rejects the selection, returns to its departure pose, and retries with an alternative candidate location.

Observed flight evaluation. A backtracking judge J^{B}, implemented by an MLLM, scores how well the observed flight V^{obs} matches the instruction segment s_i:

$$z^{\mathrm{obs}} = J^{\mathrm{B}}\big(s_i, V^{\mathrm{obs}}\big) \in [0, 1]. \tag{3}$$

When $z^{\mathrm{obs}} \geq \tau$, the location is accepted and navigation proceeds to the next segment s_{i+1}; otherwise, the selection is revised as described below.

Revised alternative candidate selection. If $z^{\mathrm{obs}} < \tau$, the UAV returns to its departure pose q^{dep} and rejects the selected target g^{F}.[1] It then selects the highest-scoring remaining candidate from $\widetilde{\mathcal{C}}_i$ using the Foresight scores $\widehat{z}(g)$ computed before departure, without generating new predicted flights. We allow at most one retry per segment; after flying to the alternative target, the UAV proceeds to the next segment without another Backtracking evaluation.

4 Experiments

4.1 Experimental Setup

Evaluation settings. We evaluate FaB-Fly on AerialVLN-S (Liu et al., 2023), using its val-seen and val-unseen splits, and on AerialVLN-Fine (Shao et al., 2026), which refines the instruction annotations of AerialVLN. Following prior work (Liu et al., 2023; Ning et al., 2026; Shao et al., 2026), we report Success Rate (SR) and Oracle Success Rate (OSR), which measure whether the UAV stops near the goal and whether it passes near the goal at any point, respectively; Navigation Error (NE), the distance between the stopping point and the goal; and nDTW and sDTW, which measure how closely the executed trajectory follows the reference trajectory, with sDTW counting only successful episodes. Trajectory-based metrics include return flights made by Backtracking. All analyses in Section 4.3 are conducted on AerialVLN-Fine.

[1] When $\widetilde{\mathcal{C}}_i$ contains only one location, no alternative exists; Backtracking is then skipped and g^{F} is accepted.

Table 1: Overall performance comparison on AerialVLN-S. **Bold** and underline indicate the first and second best results.

Category		Method	Val-Seen				Val-Unseen			
			SR ↑	OSR ↑	sDTW ↑	NE ↓	SR ↑	OSR ↑	sDTW ↑	NE ↓
Statistical		Random	0.0	0.0	0.0	109.6	0.0	0.0	0.0	149.7
		Action Sampling	0.9	5.7	0.3	213.8	0.2	1.1	0.1	237.6
Learning-Based		Seq2Seq	4.8	19.8	1.6	146.0	2.3	11.7	0.7	218.9
		CMA	3.0	23.2	0.6	121.0	3.2	16.0	1.1	172.1
		LAG	7.2	15.7	2.4	90.2	5.1	10.5	1.4	127.9
Zero-Shot	Indoor	NavGPT	0.0	0.0	0.0	163.5	0.0	0.0	0.0	82.1
		MapGPT	2.1	4.7	0.8	124.9	0.0	0.0	0.0	107.0
	Aerial	STMR	12.6	**31.6**	–	96.3	10.8	23.0	–	119.5
		CityNavAgent	13.9	30.2	5.1	80.8	11.7	35.2	**5.0**	**60.2**
		LookasideVLN	14.7	31.2	5.4	**77.1**	12.6	**36.0**	3.6	100.9
		FaB-Fly (Ours)	**18.6**	30.9	**5.6**	79.3	**15.3**	28.3	4.2	79.7

Table 2: Performance on AerialVLN-Fine.

Method	SR ↑	nDTW ↑	NE ↓
NavGPT	0.3	15.9	110.9
DiscussNav	2.6	19.6	98.4
FineCog-Nav	8.0	**22.5**	**91.4**
FaB-Fly (Ours)	**15.1**	9.1	97.6

Table 3: Ablation of FaB-Fly's components.

Method	SR ↑	nDTW ↑	NE ↓
w/o Both	14.3	10.4	116.3
w/o Foresight	16.3	9.4	111.4
w/o Backtracking	15.0	10.5	111.3
FaB-Fly	15.1	9.1	97.6

Baselines. We compare FaB-Fly with baselines from three categories. *Statistical* baselines select actions at random or by sampling from the action distribution of the training split (Liu et al., 2023). *Learning-based* baselines are trained on the AerialVLN-S training split (Anderson et al., 2018; Krantz et al., 2020; Liu et al., 2023). *Zero-shot* baselines use LLMs or MLLMs without task-specific training, including methods designed for indoor VLN (Zhou et al., 2024; Chen et al., 2024; Long et al., 2024) and for aerial VLN (Gao et al., 2025; Zhang et al., 2025a; Shao et al., 2026; Ning et al., 2026). FaB-Fly also falls into this category, as it requires no task-specific training. Among them, CityNavAgent and LookasideVLN, as well as FaB-Fly, use prior environmental data (e.g., landmark memory).

Implementation details. We use Gemini 3.1 Pro (Gemini Team, 2026) as the foresight and backtracking judges, Wan-3.0-Prime (Wan Team, 2025) as the video world model, and Qwen 3.8 27B (Qwen Team, 2025) for landmark description, instruction grounding, and low-level control. Following Ning et al. (2026), we construct the landmark memory from images along trajectories sampled from the training split for val-seen, and from images captured across each scene for val-unseen and AerialVLN-Fine. The length of each predicted flight video is scaled between 5 and 10 s according to the distance and turning angle to the candidate location.

4.2 Main Results

Results on AerialVLN-S. Table 1 compares FaB-Fly with existing methods on AerialVLN-S. FaB-Fly achieves the highest SR on both val-seen and val-unseen, and ranks first or second on six of the eight metrics. Compared with LookasideVLN, the strongest baseline, which also retrieves candidate locations from a landmark memory, FaB-Fly improves SR by 3.9 and 2.7 points on val-seen and val-unseen, respectively. It also achieves the highest sDTW on val-seen and the second highest on val-unseen. These results suggests the effectiveness of assessing candidate flights against the instruction segment both before and after execution.

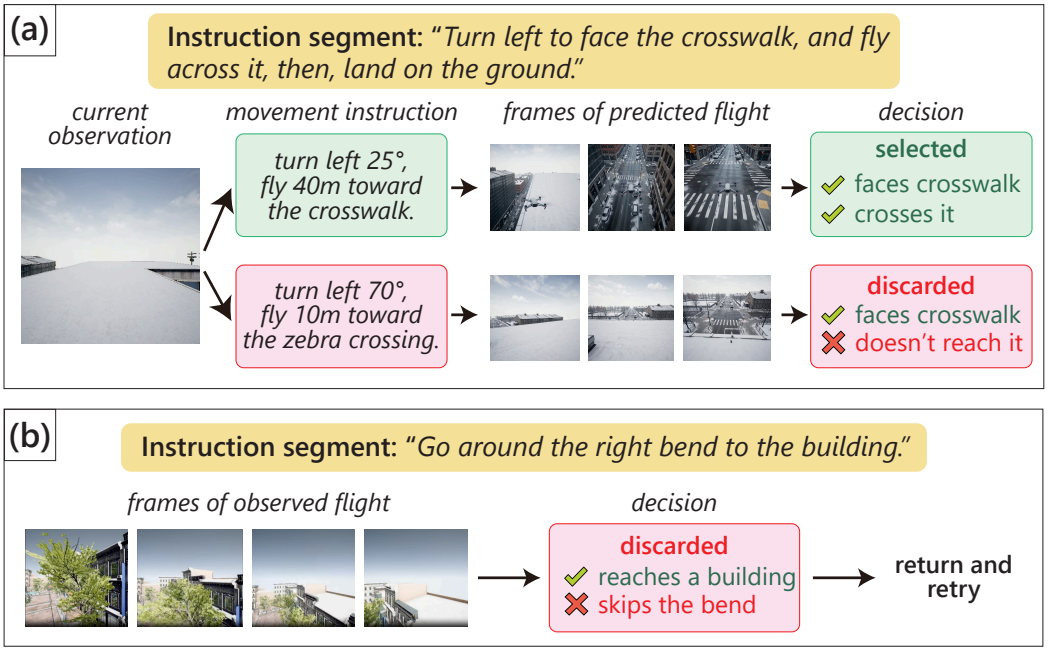

Figure 3: Qualitative examples. (a) Foresight: the predicted flights reveal that only one candidate is reached as instructed. (b) Backtracking: the observed flight shows that the UAV skipped the bend, allowing FaB-Fly to detect the incorrect selection.

Table 4: Ablation of the judge inputs.

Foresight	Backtracking	SR ↑	nDTW ↑	NE ↓
Image	Image	14.4	7.8	102.6
Image	Video	13.0	9.5	93.2
Video	Image	11.7	8.0	89.8
Video	Video	15.1	9.1	97.6

Table 5: Effect of the judge MLLM.

Judge	SR ↑	nDTW ↑	NE ↓
None (w/o Both)	14.3	10.4	116.3
Gemma 4 26B A4B	15.0	10.2	92.3
Qwen 3.8 27B	15.7	10.2	107.1
Gemini 3.1 Pro (default)	15.1	9.1	97.6

Results on AerialVLN-Fine. As shown in Table 2, FaB-Fly also achieves the highest SR on AerialVLN-Fine, improving it by 7.1 points over FineCog-Nav, with the second-best NE. Its lower nDTW is likely because the low-level controller flies directly toward each selected landmark location rather than tracing the finer shape of the reference path.

4.3 Analysis

Effect of Foresight and Backtracking. Table 3 reports the results of removing Foresight (selecting candidates without predicted flights), Backtracking (accepting each selection without evaluating the observed flight), or both. Although *w/o Foresight* achieves the highest SR, combining the two components reduces NE by 18.7 from *w/o Both*, far more than either component alone (5.0 and 4.9). The variants with Backtracking show lower nDTW, as expected, because trajectory-based metrics include return flights.

Effect of flight videos for the judges. To examine whether flight videos help the judges, we replace the video input of each judge with images (Table 4). For Foresight, "Image" scores each candidate based on the current observation and the candidate's location information, without predicting a flight video. For Backtracking, "Image" evaluates the flight only from the observations at departure and on arrival. Using videos for both judges improves all metrics over using images for both. Replacing the video of only one judge reduces NE but lowers SR even below the image-

only variant. This suggests that flight videos are most effective for SR when both judges use them, consistent with the complementarity of Foresight and Backtracking.

Effect of the judge MLLM. Table 5 compares different MLLMs used for both the foresight and backtracking judges. All judges improve over *w/o Both* in both SR and NE, including the open-weight Gemma 4 26B A4B (Gemma Team, 2026) and Qwen 3.8 27B. This suggests that the gains of FaB-Fly do not depend on a specific proprietary model, and that MLLMs with reasonable video understanding capability can serve as the judges.

Qualitative examples. Figure 3 shows examples of Foresight and Backtracking. In Figure 3(a), the UAV starts on a rooftop, and its current observation shows little beyond the roof and the sky. Two crosswalks are retrieved as candidates, and choosing between them from this view requires inferring that a 10 m flight cannot reach the second one. The predicted flights make this evident: the flight toward the first candidate reaches and crosses the crosswalk, whereas the flight toward the second stops short of it. As a result, FaB-Fly successfully selects the correct crosswalk. In Figure 3(b), the UAV is instructed to go around the right bend to the building, but it flies straight and reaches a different building. The arrival view alone does not reveal this error, since the reached building matches the description, but the observed flight shows that the UAV skipped the bend. The backtracking judge therefore rejects the selection, and the UAV returns to its departure point.

Computational cost. Foresight and Backtracking increase computation through video prediction, MLLM evaluation, and return flights. To examine whether this increase is justified by the performance gain, we compare FaB-Fly with *Visit-All*, a more costly strategy that is more likely to find the correct candidate. For each segment, Visit-All flies to every narrowed candidate and back, evaluates each observed flight with the backtracking judge, and finally flies to the best-scored candidate. Figure 4 plots SR against the computation time per episode, including flight time. FaB-Fly increases the time by only $\times 1.20$ over *w/o Both* while improving SR by 0.8 points. Visit-All achieves a slightly higher SR (+0.7 points over FaB-Fly), but requires $\times 1.86$ the time of FaB-Fly because it physically visits every candidate. This result indicates that FaB-Fly offers a favorable trade-off between cost and performance.

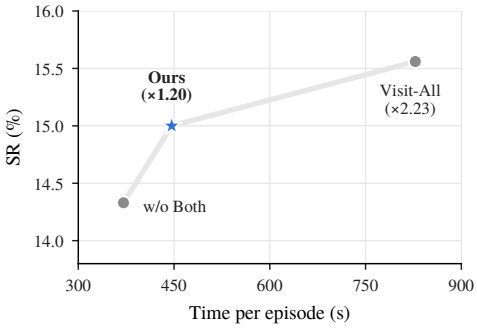

Figure 4: Success rate versus computation per episode.

5 Conclusion

We presented FaB-Fly, a zero-shot hierarchical framework for Aerial VLN that assesses flights against the current instruction segment both before and after execution. Foresight predicts the flight to each candidate location with a video world model and selects the target whose predicted flight best matches the instruction segment. Backtracking evaluates the observed flight after execution and, when it is inconsistent with the segment, returns the UAV to its departure point to retry with an alternative candidate. Experiments on AerialVLN-S and AerialVLN-Fine showed that FaB-Fly outperforms the strongest baselines in success rate, and ablation studies showed that Foresight and Backtracking work complementarily. Further analyses showed that FaB-Fly achieves success rates close to exhaustively visiting every candidate at a much lower cost.

Limitation and future directions. First, FaB-Fly requires more computation than flying directly to the selected target, since it generates a video for each candidate, queries the MLLM judges, and makes return flights. Although it offers a favorable trade-off between cost and performance (Section 4.3), reducing the cost more fundamentally would require incorporating the capabilities of the world model and the judges into the navigation model itself, rather than connecting them as separate modules. Second, because the hierarchical design flies toward landmark locations, the executed trajectory can deviate from the finer shape of the reference path, as reflected in the lower nDTW on AerialVLN-Fine. Future work could retain landmark locations as subgoals while having the high-level planner and the low-level controller cooperate more closely in deciding how to move toward them.

AI USE STATEMENT

In this work, we used generative AI tools to help develop the conceptual framework, provide feedback on the research methodology and experiments, including the design of Foresight and Backtracking and of the ablation studies, implement parts of the method and experiments, assist with translation, and interpret results. We have not used generative AI tools to propose or refine hypotheses, clean or reformat datasets, or support qualitative data analysis, and generating synthetic data sets, developing theoretical models, and formulating or proving mathematical claims are not applicable to this work. Additionally, we used generative AI tools to create and modify figures, edit and debug code, draft parts of the paper, improve its readability, suggest its structure and title, identify, search for, and summarize related literature, format references, and brainstorm research directions. We have reviewed all AI-assisted work. Specifically, LLM-generated code was reviewed by the authors and verified by inspecting intermediate outputs such as predicted flight videos, judge scores, and trajectories; all cited references were checked against the original papers; and all reported numbers were checked against the experimental logs. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

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 *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. URL https://arxiv.org/abs/1711.07280.

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. URL https://arxiv.org/abs/2606.13494.

Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. URL https://arxiv.org/abs/2412.03572.

Jiaqi Chen, Bingqian Lin, Ran Xu, Zhenhua Chai, Xiaodan Liang, and Kwan-Yee K. Wong. MapGPT: Map-guided prompting with adaptive path planning for vision-and-language navigation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

Kehan Chen, Dong An, Yan Huang, Rongtao Xu, Yifei Su, Yonggen Ling, Ian Reid, and Liang Wang. Constraint-aware zero-shot vision-language navigation in continuous environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 47(11):10441–10456, 2025. URL https://arxiv.org/abs/2412.10137.

Yifei Dong, Fengyi Wu, Guangyu Chen, Lingdong Kong, Qiyu Hu, Yuxuan Zhou, Xu Zhu, Jingdong Sun, Jun-Yan He, Qi Dai, Alexander G. Hauptmann, and Zhi-Qi Cheng. Towards unified world models for visual navigation via memory-augmented planning and foresight. In *Proc. European Conference on Computer Vision (ECCV)*, 2026. URL https://arxiv.org/abs/2510.08713.

Yunpeng Gao, Zhigang Wang, Pengfei Han, Linglin Jing, Dong Wang, and Bin Zhao. Exploring spatial representation to enhance LLM reasoning in aerial vision-language navigation. *arXiv preprint arXiv:2410.08500v3*, 2025. URL https://arxiv.org/abs/2410.08500v3.

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. In *Proc. International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2502.18041.

Gemini Team. Gemini 3.1 pro: Technical documentation and model overview. https://deepmind.google/technologies/gemini/, 2026. Accessed: 2026-09-26.

Gemma Team. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Xijie Huang, Weiqi Gai, Tianyue Wu, Congyu Wang, Zhiyang Liu, Xin Zhou, Yuze Wu, and Fei Gao. NavDreamer: Video models as zero-shot 3D navigators. *arXiv preprint arXiv:2602.09765*, 2026. URL https://arxiv.org/abs/2602.09765.

Yanjia Huang, Xianshun Jiang, Xiangbo Gao, Mingyang Wu, and Zhengzhong Tu. VISTAv2: World imagination for indoor vision-and-language navigation. *arXiv preprint arXiv:2512.00041*, 2025. URL https://arxiv.org/abs/2512.00041.

Liyiming Ke, Xiujun Li, Yonatan Bisk, Ari Holtzman, Zhe Gan, Jingjing Liu, Jianfeng Gao, Yejin Choi, and Siddhartha Srinivasa. Tactical rewind: Self-correction via backtracking in vision-and-language navigation. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. URL https://arxiv.org/abs/1903.02547.

Jing Yu Koh, Honglak Lee, Yinfei Yang, Jason Baldridge, and Peter Anderson. Pathdreamer: A world model for indoor navigation. In *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. URL https://arxiv.org/abs/2105.08756.

Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *Proc. European Conference on Computer Vision (ECCV)*, 2020.

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 *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025. URL https://arxiv.org/abs/2406.14240.

Shaoxuan Li, Xiangyu Dong, Xiaoguang Ma, Junfeng Chen, Haoran Zhao, and Yaoming Zhou. CLOSER-VLN: Closed-loop self-verified retrieval-augmented reasoning for aerial vision-language navigation. *arXiv preprint arXiv:2606.28397*, 2026. URL https://arxiv.org/abs/2606.28397.

Shubo Liu, Hongsheng Zhang, Yuankai Qi, Peng Wang, Yanning Zhang, and Qi Wu. AerialVLN: Vision-and-language navigation for UAVs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Liu_AerialVLN_Vision-and-Language_Navigation_for_UAVs_ICCV_2023_paper.html.

Xuchen Liu, Jiawei Huang, Shihao Xia, Bingxi Liu, Jinqiang Cui, and Jiankun Yang. ImagineUAV: Aerial vision-language navigation via world-action modeling and kinodynamic planning. *arXiv preprint arXiv:2606.01205*, 2026a. URL https://arxiv.org/abs/2606.01205.

Yutong Liu, Xiaojie Li, Mingzhu Xu, and Jianlong Wu. AirForesight: Current-to-future spatial map imagination with cross-space planning consistency for UAV-VLN. In *Proceedings of the 34th ACM International Conference on Multimedia*, 2026b. doi: 10.1145/3767308.3836065. URL https://arxiv.org/abs/2608.12835.

Yuxing Long, Xiaoqi Li, Wenzhe Cai, and Hao Dong. Discuss before moving: Visual language navigation via multi-expert discussions. In *IEEE International Conference on Robotics and Automation*, 2024. URL https://ieeexplore.ieee.org/document/10611565/.

Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. Self-monitoring navigation agent via auxiliary progress estimation. In *International Conference on Learning Representations*, 2019a. URL https://openreview.net/forum?id=r1GAsjC5Fm.

Chih-Yao Ma, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira. The regretful agent: Heuristic-aided navigation through progress estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019b. URL https://arxiv.org/abs/1903.01602.

Yanghong Mei, Longteng Guo, Ming-Ming Yu, Guiyu Zhao, Xingjian He, and Jing Liu. NavWM: A unified navigation world model for foresight-driven planning. In *Proc. European Conference on Computer Vision (ECCV)*, 2026. URL https://arxiv.org/abs/2606.24101.

Yuwei Ning, Ganlong Zhao, Yipeng Qin, Si Liu, Yang Liu, Liang Lin, and Guanbin Li. Lookaside-VLN: Direction-aware aerial vision-and-language navigation. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Dhruv Shah, Błażej Osiński, Brian Ichter, and Sergey Levine. LM-Nav: Robotic navigation with large pre-trained models of language, vision, and action. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 492–504. PMLR, 2023. URL https://proceedings.mlr.press/v205/shah23b.html.

Dian Shao, Zhengzheng Xu, Peiyang Wang, Like Liu, Yule Wang, Jieqi Shi, and Jing Huo. FineCog-Nav: Integrating fine-grained cognitive modules for zero-shot multimodal UAV navigation. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pp. 1325–1334, June 2026. URL https://arxiv.org/abs/2604.16298.

Xiangyu Shi, Zerui Li, Wenqi Lyu, Jiatong Xia, Feras Dayoub, Yanyuan Qiao, and Qi Wu. Smart-Way: Enhanced Waypoint Prediction and Backtracking for Zero-Shot Vision-and-Language Navigation. In *Proc. International Conference on Intelligent Robots and Systems (IROS)*, 2025. URL https://arxiv.org/abs/2503.10069.

Wan Team. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Boxiong Wang, Hui Kang, Geng Sun, Jiahui Li, Chao Yu, and Daxin Tian. RecoverFly: A failure-aware reinforcement learning post-training framework for aerial vision-language navigation. *arXiv preprint arXiv:2608.09467*, 2026a. URL https://arxiv.org/abs/2608.09467.

Hanqing Wang, Wei Liang, Luc Van Gool, and Wenguan Wang. DREAMWALKER: Mental planning for continuous vision-language navigation. In *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. URL https://arxiv.org/abs/2308.07498.

Xiangyu Wang, Donglin Yang, Ziqin Wang, Hohin Kwan, Jinyu Chen, Wenjun Wu, Hongsheng Li, Yue Liao, and Si Liu. Towards realistic UAV vision-language navigation: Platform, benchmark, and methodology. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/15ce8e7afe5ee95bad56e3b9be28d3d1-Abstract-Conference.html.

Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. In *Proc. European Conference on Computer Vision (ECCV)*, 2018. URL https://arxiv.org/abs/1803.07729.

Yunheng Wang, Yuetong Fang, Taowen Wang, Yixiao Feng, Yawen Tan, Shuning Zhang, Peiran Liu, Yiding Ji, and Renjing Xu. DreamNav: A trajectory-based imaginative framework for zero-shot vision-and-language navigation. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2026b.

Zihan Wang, Xiangyang Li, Jiahao Yang, Yeqi Liu, Junjie Hu, Ming Jiang, and Shuqiang Jiang. Lookahead exploration with neural radiance representation for continuous vision-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13753–13762, 2024. URL https://arxiv.org/abs/2404.01943.

Haotian Xu, Yue Hu, Chen Gao, Zhengqiu Zhu, Yong Zhao, and Quanjun Yin. GeoNav: Empowering MLLMs with dual-scale geospatial reasoning for language-goal aerial navigation. *Pattern Recognition*, 177(C), September 2026. ISSN 0031-3203. doi: 10.1016/j.patcog.2026.113365. URL https://doi.org/10.1016/j.patcog.2026.113365.

Xuan Yao, Junyu Gao, and Changsheng Xu. NavMorph: A self-evolving world model for vision-and-language navigation in continuous environments. In *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5536–5546, 2025. URL https://arxiv.org/abs/2506.23468.

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 *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025a.

Weichen Zhang, Peizhi Tang, Xin Zeng, Fanhang Man, Shiquan Yu, Zichao Dai, Baining Zhao, Hongjin Chen, Yu Shang, Wei Wu, Chen Gao, Xinlei Chen, Xin Wang, Yong Li, and Wenwu Zhu. Aerial world model for long-horizon visual generation and navigation in 3D space. *arXiv preprint arXiv:2512.21887*, 2025b. URL https://arxiv.org/abs/2512.21887.

Baining Zhao, Jiacheng Xu, Weicheng Feng, Xin Zhang, Zhaolu Wang, Haoyang Wang, Shilong Ji, Ziyou Wang, Jianjie Fang, Zhiheng Zheng, Weichen Zhang, Yu Shang, Wei Wu, Chen Gao, Xinlei Chen, and Yong Li. WorldVLN: Autoregressive world action model for aerial vision-language navigation. *arXiv preprint arXiv:2605.15964*, 2026. URL https://arxiv.org/abs/2605.15964.

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*, 2026. URL https://arxiv.org/abs/2606.06147.

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*, 2024. URL https://arxiv.org/abs/2305.16986.

A Framework Details

Our navigation framework follows prior memory-based hierarchical approaches (Zhang et al., 2025a; Ning et al., 2026) in landmark memory construction, instruction grounding, and candidate narrowing. This section describes each component in detail.

A.1 Landmark Memory Construction

The landmark memory is built from RGB-D images collected in the prior survey. First, an MLLM describes the landmarks visible in each image and marks a representative image point for each of them. Then, the three-dimensional location of each landmark is recovered in world coordinates by back-projecting its image point using the depth value, the camera intrinsics, and the known camera pose. Finally, detections of the same landmark from different images are merged into a single entry based on their spatial proximity. The result is the landmark memory $\mathcal{M} = \{(d_n, g_n)\}_{n=1}^{N}$, where d_n is a landmark description and $g_n \in \mathbb{R}^3$ is its estimated location in world coordinates.

A.2 Instruction Grounding

An MLLM parses the instruction x into K ordered segments $s_1, \ldots, s_K$, one for each landmark the instruction refers to. From each segment s_i, such as "turn left to approach the gas station," the MLLM extracts a landmark phrase ℓ_i ("gas station") and a motion phrase m_i that leads the UAV to that landmark ("turn left"). When no motion can be identified for ℓ_i, the segment has no motion phrase. For each segment, the memory entries whose descriptions d_n (e.g., "petrol station") have a text-embedding similarity to ℓ_i above a threshold are retrieved. The locations of the retrieved entries form the set of candidate locations $\mathcal{C}_i \subseteq \{g_1, \ldots, g_N\}$ for that segment.

A.3 Candidate Narrowing

The segments are processed one at a time, in instruction order. When the flights for $s_1, \ldots, s_{i-1}$ have been completed, the UAV is at its current position p (or at the starting position when $i = 1$), and the planner must choose the location to visit next from $\mathcal{C}_i$. Passing all of $\mathcal{C}_i$ to the planner would be inefficient, since a landmark phrase often matches many memory entries and the planner generates and judges a video for each candidate. It would also be insufficient, since a location is correct only if the remaining segments can also be followed from it. We therefore narrow $\mathcal{C}_i$ with a lightweight geometric heuristic that evaluates the entire remaining route.

Route cost. A candidate route is a sequence $r = (r_i, \ldots, r_K)$ with $r_j \in \mathcal{C}_j$. Writing $r_{i-1} = p$, the flight for segment s_j moves the UAV by the displacement $\Delta_j(r) = r_j - r_{j-1}$. Approximating each flight by a straight segment, the UAV arrives at r_{j-1} heading along $\Delta_{j-1}(r)$, so reaching r_j requires turning by the angle $\theta_j(r)$ from $\Delta_{j-1}(r)$ to $\Delta_j(r)$ in the horizontal plane. Each motion phrase m_j is mapped to a reference turn angle $\phi(m_j)$, for example $\phi(\text{"turn left"}) = +45^\circ$ and $\phi(\text{"turn right"}) = -45^\circ$. A route is penalized when its turns deviate from these reference angles and when it travels far:

$$\Gamma(r) = \sum_{j=i}^{K} \Big(\big|\theta_j(r) - \phi(m_j)\big| \; + \; \lambda \, \|\Delta_j(r)\| \Big), \tag{4}$$

where the angular difference is wrapped to $[0^\circ, 180^\circ]$ and λ balances the two terms. The turn term is omitted for a segment without a motion phrase, and for the first segment, the current heading of the UAV is used in place of $\Delta_{i-1}(r)$.

Route search. Since the number of candidate routes grows exponentially with the number of remaining segments, we search them by beam search with a beam width of 128, extending partial routes one segment at a time in instruction order and ranking them by their accumulated cost. Among the complete routes, the eight with the lowest cost are retained, and a route is skipped when its final location lies within 10 m of that of an already retained route, so that the retained routes lead to distinct destinations. The first locations of the retained routes form $\widetilde{\mathcal{C}}_i$, which therefore contains at most eight locations. This narrowing is recomputed from the current position at every segment, so

the locations for later segments are not fixed in advance. In the *w/o Foresight* variant (Section 4.3), the UAV flies to the first location of the lowest-cost route.