AHEADFLY: THINK AND ACT AHEAD FOR MULTI-STAGE AERIAL VISION-AND-LANGUAGE NAVIGATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Aerial vision-and-language navigation (AVLN) requires unmanned aerial vehicles (UAVs) to follow natural-language instructions in open 3D environments. Multi-stage navigation requires sequentially grounding multiple landmarks. Existing zero-shot approaches treat each stage as an independent grounding problem and primarily address what to look at in each stage. However, we argue that the dynamic nature of UAVs introduces additional challenges: aerial navigation must also resolve "where to look" and "where to look from" while maintaining grounding continuity across stages. In this paper, we propose **AheadFly**, a training-free modular framework that thinks and acts ahead for multi-stage aerial navigation. First, it "**thinks ahead**" by parsing instructions into explicit landmark queries and inter-stage transition directions. It then "**acts ahead**" by jointly adjusting the UAV's orientation and flight path under the guidance of these directions. Specifically, a pre-grounding view adjustment decides where to look by reorienting the UAV so the current landmark enters its field of view, and a lookahead path adjustment decides where to look from through multiple guiding forces, to mitigate cross-stage landmark occlusion and avoid obstacles along the path. To evaluate intermediate grounding and continuity, we introduce **LandmarkFly**, a multi-stage, long-range AVLN benchmark generated through a landmark-centric pipeline with explicit instruction-trajectory-landmark alignment, together with stage-level evaluation metrics. Experiments on both OpenFly-AirSim and LandmarkFly show consistent gains over the strongest baselines, improving Success Rate by 5.5 and 32.4 percentage points on the two benchmarks, respectively, and Grounding Point Accuracy by 7.6 percentage points on LandmarkFly.

1 INTRODUCTION

Vision-Language Navigation (VLN) is a foundational embodied AI task where agents follow natural language instructions to navigate physical environments. In low-altitude environments, aerial VLN (AVLN) enables UAVs to navigate to designated targets from natural-language instructions under obstacle avoidance and other dynamic constraints (Liu et al., 2023; Peng et al., 2025). AVLN serves tasks such as aerial inspection, emergency response, and urban management, where a UAV must ground and then reach multiple described landmarks in sequence (Zhao et al., 2026). Compared with ground navigation, AVLN covers a larger 3D workspace with broader visual coverage, and how a landmark appears there depends on the UAV's altitude and observation pose, unlike ground navigation moving on a roughly fixed 2D plane. Early AVLN studies transferred established ground-based VLN methods to aerial settings, but their effectiveness was limited by larger scenes, freer 3D motion, and weaker environmental structure (Yao et al., 2025; Lyu et al., 2026). Later policy learning-based end-to-end methods learned mappings from language and vision to flight actions, but required task-specific data and training and offered limited interpretability (Sautenkov et al., 2025; Wu et al., 2025; Gao et al., 2026; Xu et al., 2026a; Wang et al., 2026b). Recent zero-shot modular methods use pretrained foundation models for instruction parsing, visual grounding, and motion control without AVLN-specific training (Shah et al., 2023; Zhou et al., 2024; Hu et al., 2025; Zhang et al., 2025b; Xu et al., 2026b; Qin et al., 2026).

While most zero-shot methods treat each stage as an independent grounding problem and primarily address "what to look at" in each stage, we highlight that the UAV's own actions determine whether the next landmark can be observed at all, so dynamic viewpoint shifts pose extra challenges to static

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.

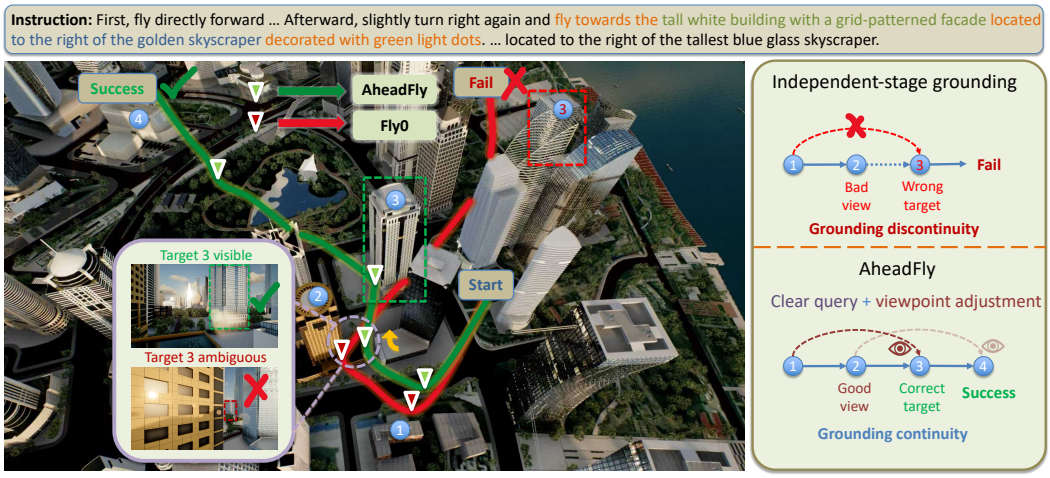

Figure 1: Motivation for thinking and acting ahead. Treating each stage as an independent grounding problem leaves the UAV with a poor view and a wrong target, breaking grounding continuity and failing the task. AheadFly supplies explicit landmark queries and adjusts the viewpoint before grounding, preserving continuity across stages.

grounding, notably relative bearing changes and landmark occlusion. These challenges require the UAV to decide **"where to look"** and **"where to look from"** before grounding. Such impacts are amplified in multi-stage AVLN tasks, where current landmarks must be grounded from the previous stage's terminal pose. Treating different stages independently leaves the UAV uncertain about where to look and where to look from after each transition (Fig. 1). Any intermediate deviation can therefore break grounding continuity in the instructed order and cause task failure. Our quantitative analysis shows a strong positive correlation between intermediate landmark grounding accuracy and overall navigation success, highlighting the need to maintain continuity across stage transitions. Current methods based on stage verification and action backtracking repair a failure only after it occurs, and thus cannot ensure cross-stage landmark visibility (Song et al., 2025; Li et al., 2025; Zheng et al., 2026b). *How to proactively preserve grounding continuity before a deviation occurs for multi-stage AVLN remains a challenging task.*

To fill this gap, we propose **AheadFly**, a training-free modular framework for aerial vision-and-language navigation. AheadFly first organizes the language instruction into a text-based Task Planning Graph (TPG), thereby **thinks ahead** about how the task should be divided into stages and how consecutive stages should be connected, and thereby supplies the information on which the subsequent actions depend. Within the TPG, explicit landmark queries obtained through instruction segmentation, coreference resolution, and target entity identification specify "what to look at", reducing grounding errors caused by semantic ambiguity and their downstream effects. The inter-stage transition directions then guide a View-Aware Landmark Grounding Module and a Multi-Force-Guided Motion Controller to **act ahead** along two axes while satisfying basic obstacle-avoidance constraints: 1) the View-Aware Landmark Grounding Module applies **pre-grounding view adjustment**, acting ahead of grounding, to reorient the UAV so that the current landmark enters the field of view whenever possible, determining "where to look" at the current stage and compensating for relative bearing changes; 2) the Multi-Force-Guided Motion Controller applies **lookahead path adjustment**, acting one stage ahead, to bias the trajectory toward the approximate bearing of the next landmark while keeping the endpoint near the current target, determining "where to look from" when grounding begins in the next stage and mitigating landmark occlusion. Together, thinking ahead and acting ahead enable AheadFly to proactively mitigate viewpoint-change challenges and maintain grounding continuity across stages.

To evaluate intermediate grounding and continuity, we introduce **LandmarkFly**, a multi-stage AVLN benchmark with stage-level instruction–trajectory–landmark alignment. We observe that final navigation outcomes alone cannot reveal when and why grounding continuity breaks down during flight. Existing AVLN benchmarks score complete trajectories by task success rate, revealing little about whether the UAV can see, ground, and approach each instructed landmark in order (Liu et al., 2023; Wang et al., 2025; Gao et al., 2026). In contrast, the landmark-centric generation procedure in our LandmarkFly benchmark organizes navigation around explicit target landmarks,

producing the trajectory and instruction while preserving fine-grained correspondence among instruction stages, target landmarks, and trajectory segments. Based on this, we design stage-level metrics for landmark visibility, grounding accuracy, and navigation progress through the instructed sequence. We evaluate our AheadFly on both OpenFly and LandmarkFly benchmarks. The results show that AheadFly achieves the best overall success rate and intermediate stage-level grounding accuracy over the strongest baselines.

Our main contributions are as follows:

- We identify two questions that dynamic viewpoint changes raise for zero-shot methods applying conventional grounding directly: "where to look" and "where to look from", both essential to grounding continuity in multi-stage tasks.
- We propose **AheadFly**, a training-free modular framework that thinks ahead by resolving each stage's requirements before flight and acts ahead on two horizons: before grounding to counter bearing changes, and one stage early to mitigate landmark occlusion.
- We construct **LandmarkFly**, an AVLN benchmark that aligns instruction stages, flight trajectories, and target landmarks, with metrics for landmark visibility, grounding accuracy, and stage-level navigation progress.

2 Related Work

2.1 Aerial Vision-and-Language Navigation

End-to-end AVLN. Aerial VLN follows the end-to-end policy-learning line established for ground VLN, from sequence-to-sequence agents (Anderson et al., 2018; Fried et al., 2018) and the pretraining-then-finetuning paradigm (Hao et al., 2020) to attention and reinforcement-learning objectives for cross-modal alignment (Hong et al., 2021; Wang et al., 2019). In aerial navigation, AerialVLN (Liu et al., 2023) defined the task and introduced the cross-modal attention baseline CMA; later work applies reinforcement-learning-based fine-tuning to particular scenes or tasks (Zhou et al., 2026) or, with foundation models and parameter-efficient fine-tuning such as LoRA, integrates history, bird's-eye-view (BEV) features, or geometry for spatial reasoning (Wang et al., 2025; Jiang et al., 2025; 2026). Despite strong performance on specific benchmarks, these methods remain black boxes, and their reliance on large-scale instruction–trajectory fine-tuning may favor route imitation over reasoning, degrading generalization on unseen scenes and routes.

Modular AVLN. Zero-shot modular methods remove the need for such training. They prompt pretrained models for semantic understanding and visual grounding while delegating low-level control to separate modules (Liang et al., 2023; Zhou et al., 2024; Zheng et al., 2026a; Ning et al., 2026). Others specialize in global reasoning and local planning (Yue et al., 2026; Zheng et al., 2026b) or expose functions as callable skills (Xin et al., 2026; Li et al., 2026). In weakly structured aerial environments (Yao et al., 2025; Lyu et al., 2026), VLFly (Zhang et al., 2025c) and SkyVLN (Li et al., 2025) match semantics by CLIP similarity and cross-attention, respectively, STMR (Gao et al., 2024) reasons over scene semantics and geometry, and See-Point-Fly (Hu et al., 2025) and Fly0 (Xu et al., 2026b) use adaptive depth mapping and EGO-Planner (Zhou et al., 2020), respectively, for waypoint control. These methods do not require task-specific training, offer greater interpretability, and generally apply more readily to unseen scenes and trajectories. In multi-stage tasks, however, such methods are likewise not always effective: coreference and relational descriptions can cause intermediate grounding errors or loss of the target view, and because each stage inherits the preceding stage's terminal pose, local errors can propagate into grounding discontinuities and task failure. Stage calibration and backtracking (Song et al., 2025; Li et al., 2025; Zheng et al., 2026b) correct failures only after they occur, so they cannot ensure visibility in transitions.

2.2 Aerial VLN Dataset Construction

AerialVLN (Liu et al., 2023) pairs expert simulated flights with crowdsourced video annotation, and OpenUAV (Wang et al., 2025) records human-operated 6 degrees of freedom (DoF) trajectories and uses GPT-4 to describe their targets and surroundings. OpenFly (Gao et al., 2026) contains 100k trajectories across 18 scenes, built by identifying landmarks, planning collision-free A* trajectories,

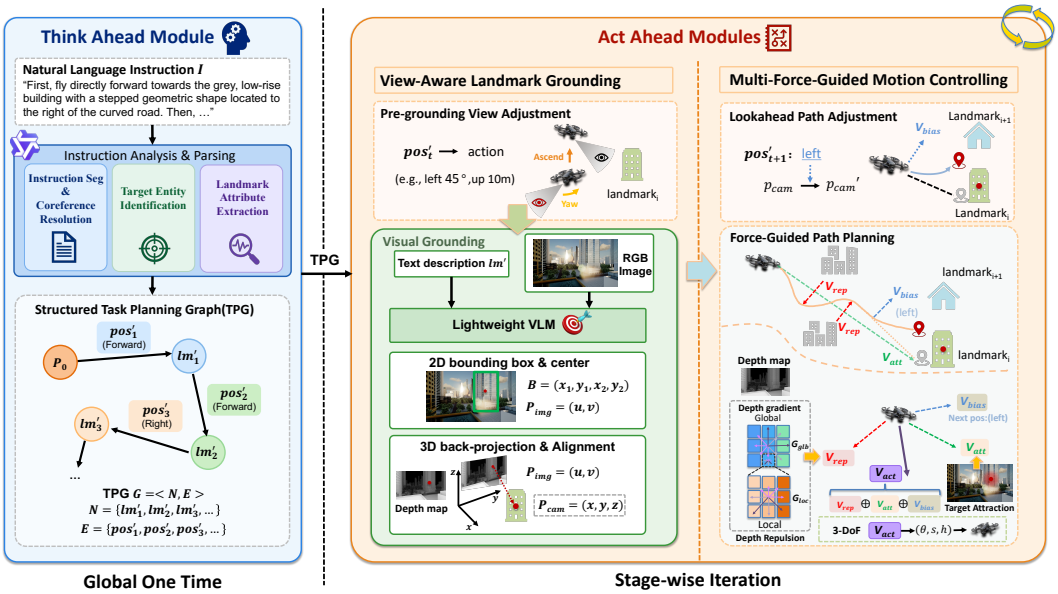

Figure 2: Overview of AheadFly. *Think ahead*: one Task Planning Graph (TPG) per instruction, with landmark queries lm'_i as nodes and transition directions pos'_i as edges. *Act ahead* per stage: pre-grounding view adjustment, VLM grounding back-projected to a 3D waypoint, then multi-force control fusing V_{att}, V_{bias} and depth-gradient V_{rep}, with lookahead path adjustment from the next direction.

and splitting them at turning points, after which a VLM produces sub-instructions that an LLM integrates. These methods first generate flight trajectories and derive natural-language instructions from them. Although this ensures valid trajectories, the resulting instructions may carry redundant stage descriptions and ambiguous landmark references, and the absence of explicit per-landmark spatial annotations makes it hard to align grounding predictions with targets and instruction stages, limiting fine-grained evaluation of grounding continuity and intermediate-stage performance.

3 METHOD

3.1 PROBLEM FORMULATION AND OVERVIEW

Problem Formulation. Given an instruction I specifying an ordered sequence of target landmarks, a UAV starts at pose $q_0 = (P_0, \psi_0)$, where $P_0 \in \mathbb{R}^3$ and ψ_0 denote its position and heading. At step t, it receives a visual observation $O_t = (R_t, D_t)$ consisting of a forward-facing RGB image and a depth map, and selects a motion action $a_t = (\Delta x_t, \Delta y_t, \Delta z_t, \Delta \psi_t)$ or STOP. Each action changes the viewpoint and thus the conditions for grounding subsequent landmarks. An episode ending at step T achieves endpoint success if $\|P_T - P_{\text{tar}}\|_2 \leq \epsilon$, where P_{tar} is the destination; intermediate performance is evaluated with stage-level metrics.

AheadFly Overview. AheadFly thinks and acts ahead to handle viewpoint changes and maintain grounding continuity across stages (Fig. 2). The task parsing module organizes instructions into a Task Planning Graph (TPG), providing landmark queries and transition directions (Section 3.2). The landmark grounding module adjusts the view before grounding and generating a 3D waypoint (Section 3.3). The motion control module approaches the current target and avoids obstacles while improving the next stage's viewing position (Section 3.4). We describe these modules below, followed by LandmarkFly's data generation method for stage-level evaluation (Section 3.5).

3.2 LLM-BASED STRUCTURED TASK PARSING

Coreferential descriptions and descriptions of non-landmark objects in long instructions can confuse grounding. For example, "turn left near A and head toward C to the right of B" contains two types of directional information: "to the right" expresses a landmark-non-landmark spatial relation that

identifies C, whereas "turn left" expresses a landmark-landmark spatial relation that guides the stage transition. Simple landmark description extraction (Zhang et al., 2025a;c; Li et al., 2025; Ning et al., 2026) or transition-based instruction segmentation (Hong et al., 2020; He et al., 2025; Gao et al., 2024; Zheng et al., 2026a) may not fully distinguish them or provide clear stage-level guidance.

The LLM-based task parser Φ_{Plan} takes the complete instruction I and divides it into stages $S = \{s_1, s_2, ..., s_M\}$. Each $s_i =< pos_i, lm_i >$ directs the UAV from near lm_{i-1} toward lm_i along relative direction pos_i, with lm_0 denoting the start. Directions are simplified to terms such as "left," "right," and "forward," while landmark descriptions undergo three operations: 1) Coreference resolution: Expand references such as "it" and "the previous building" using context, making each stage independently interpretable. 2) Target entity identification: Distinguish the landmark to approach from non-landmark reference objects. For example, B is the target in "go to B to the left of A." 3) Landmark attribute extraction: Remove irrelevant content while retaining visual attributes and identifying spatial relations. Place the latter after the target description, as in "B, to the left of A." More details are included in Appendix A, while the LLM prompts can be found in Appendix I.

The parsed stages form a text-based directed **Task Planning Graph (TPG)**, $G =< N, E >$. Nodes $N = \{lm'_1, lm'_2, ..., lm'_M\}$ represent explicit landmark queries, and edges $E = \{pos'_1, pos'_2, ..., pos'_M\}$ represent inter-stage transition directions. The queries reduce grounding errors caused by semantic ambiguity, while the directions guide viewpoint and path adjustments, jointly supporting grounding continuity.

3.3 View-Aware Landmark Grounding

The landmark grounding module Φ_{Grd} acts ahead through **pre-grounding view adjustment**. Before grounding landmark lm'_i, it converts the TPG transition direction pos'_i into a pose adjustment that improves the target's visibility to the forward-facing camera. For example, "ahead and to the left" triggers a 45° left yaw rotation, while "ahead at a higher elevation" triggers a $10\,\mathrm{m}$ ascent. These adjustments use the approximate target direction to prepare the observation before invoking the grounding VLM.

Given the refined query lm'_i and the adjusted RGB observation O_i, a lightweight grounding VLM predicts a bounding box $B = (x_1, y_1, x_2, y_2)$. Its center $P_{\text{pix}} = (u, v)$, where $u = (x_1 + x_2)/2$ and $v = (y_1 + y_2)/2$, is back-projected into the camera frame using the aligned depth map D_i, as in Fly0 (Xu et al., 2026b). The resulting point P_{cam} defines a local 3D waypoint for motion control.

As the UAV moves, a **persistence mechanism** updates the waypoint's camera-frame coordinates using relative motion, without explicitly transforming it into the world frame. Writing $P_{\text{cam}} = (x, y, z)^\top$, the update is

$$P'_{\text{cam}} = \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} x + s\sin\theta \\ y - h \\ z - s\cos\theta \end{pmatrix}, \tag{1}$$

where θ is the yaw change, and s and h denote horizontal travel distance and vertical displacement, respectively. This update maintains the waypoint across motion steps without repeated VLM grounding or explicit world-frame camera extrinsics.

3.4 Multi-Force-Guided Motion Control

Reaching the current waypoint does not ensure a suitable view for the next stage: stopping too close to the current landmark may cause occlusion that reorientation alone cannot resolve. The **Multi-Force-Guided Motion Controller (MFG)** Φ_{Act} therefore uses the next TPG direction to bias the endpoint toward the next landmark's approximate bearing while approaching the current target and satisfying obstacle-avoidance constraints. This **lookahead path adjustment** determines where to look from at the next stage and, together with pre-grounding view adjustment, mitigates the effects of viewpoint changes and improves grounding continuity.

In the image plane, the controller generates an attractive vector V_{att} toward the current waypoint and a bias vector $V_{bias} = (\delta_x \cdot |x_1 - x_2|, \delta_y \cdot |y_1 - y_2|)$ from box B and the next stage's direction pos'_{i+1}, when available. The factors δ_x, δ_y map discrete directions to horizontal and vertical offsets. Box width and height scale the offsets to remain within the landmark's visible extent, reducing

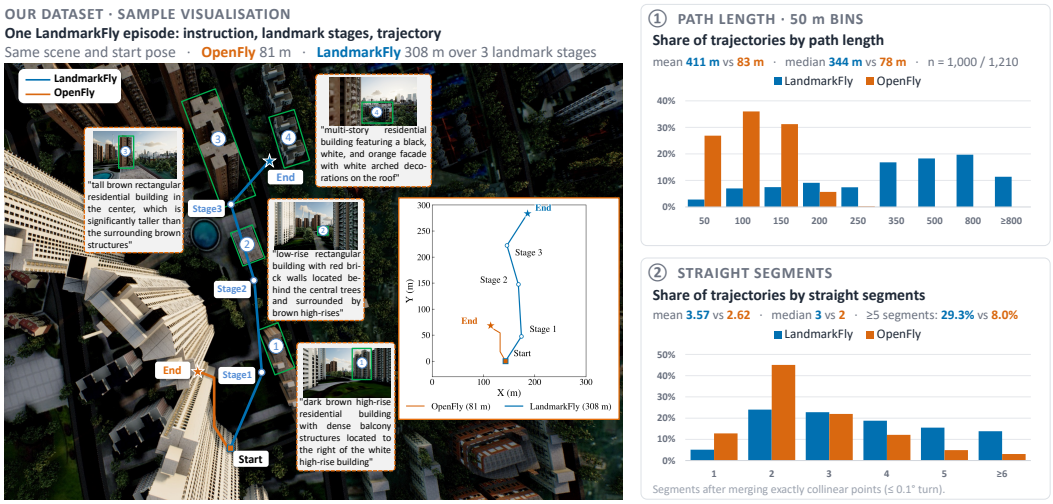

Figure 3: LandmarkFly versus OpenFly. LandmarkFly instructions name each stage's target explicitly; its trajectories are longer (mean 411 m vs. 83 m) and turn more often (3.57 vs. 2.62 straight segments; $\geq$5 in 29.3% vs. 8.0%). In the sample episode (same scene and start pose), OpenFly covers 81 m while LandmarkFly spans three landmark stages over 308 m.

occlusion of the subsequent view by the current target. The obstacle-avoidance repulsive vector V_{rep} is computed from a hierarchical depth grid. After clipping depth to 255 and average-pooling a 3×3 grid with 3×3 subcells per block, differences between surrounding and center cells yield global and local gradients. Negative gradients are transferred to their opposite directions, and maximum-scaling softmax converts the resulting positive gradients into directional weights. Weighted sums of eight 2D basis directions are combined into V_{rep}, which steers the UAV toward open space. The full formulation is given in Appendix B. Then the final action vector is

$$V_{act} = \eta \cdot Norm(V_{att} + V_{bias}) + (1 - \eta) \cdot (\delta \cdot V_{rep}), \tag{2}$$

where $Norm(\cdot)$ denotes normalization. The weight η adjusts attraction according to vector magnitude and flight phase, prioritizing waypoint alignment at stage onset and completion to retain target visibility and reach the endpoint. The weight δ attenuates repulsion in open foregrounds, reducing yaw oscillations caused by small depth changes. More details on the values of both are included in Appendix C. Finally, the endpoint of V_{act} is projected into camera coordinates $P_{act} = (x, y, z)$. The heading, horizontal distance, and vertical displacement are $\theta = arctan(\frac{x}{z})$, $s = \sqrt{x^2 + z^2}$, and $h = -y$. Step-size limits and the persistence update produce the current 3-DoF command, supporting obstacle avoidance and favorable observations at the next stage.

3.5 Landmark-Centric Dataset Construction

To assess intermediate grounding, LandmarkFly uses a fully automated **landmark-centric** data generation pipeline with **explicit stage annotations** (Fig. 3). From a sampled start P^0_{3D} and stage count N, each stage i: 1) Selects a depth-based safe viewing direction Y^i_{safe} from ahead, up, down, left, or right, reorienting the UAV to $P^i_{pos} =< P^{i-1}_{3D}, Y^i_{safe} >$. 2) Uses a VLM to select a distinctive landmark at P^i_{raw} and describe its appearance and optional spatial relations as I^i_{raw}. 3) Grounds this description near P^i_{raw} to obtain box B^i_{tar} and center P^i_{tar}, then matches visible objects' AirSim-projected 3D boxes by maximum intersection over union (IoU) to record scene identifier M^i. 4) Back-projects P^i_{tar} using depth to world anchor P^i_{3D} and flies there.

After N stages, an LLM integrates descriptions and anchors into instruction I; anchors determine trajectory T. Landmark annotations $L = \{< P^i_{pos}, B^i_{tar}, M^i >\}_{1\leq i\leq N}$ yield dataset $< I, T, L >$ with explicit landmark location annotations and multi-stage instruction-landmark-trajectory alignment. The prompts for the VLM and LLM are provided in Appendix J.

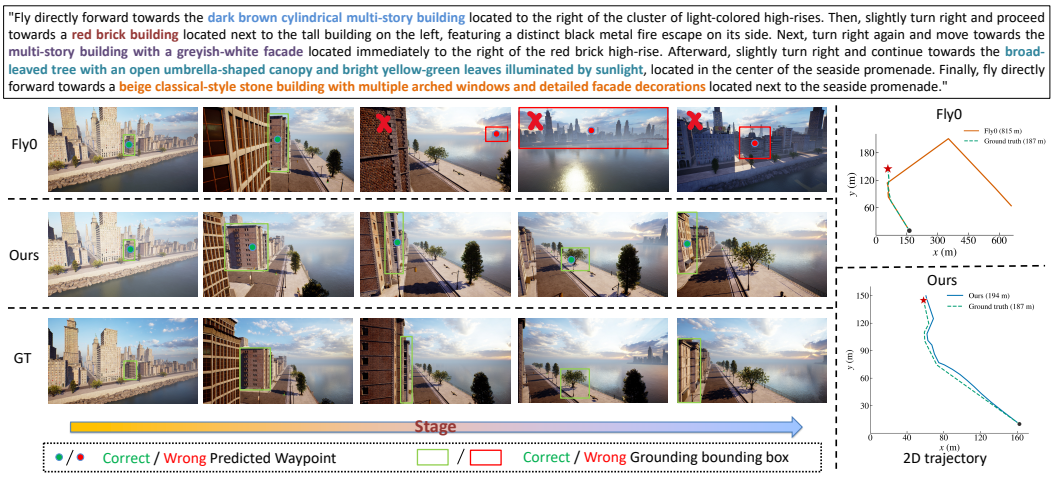

Figure 4: Qualitative comparison with Fly0 on a five-stage task. Green and red mark correct and incorrect predicted waypoints and grounding boxes. Fly0 drifts to non-instructed objects, whereas AheadFly keeps its boxes on the instructed landmarks and follows the ground-truth route.

4 Experiments

4.1 Experimental Setup

Simulation Environment and Benchmark. We use Unreal Engine 4 (UE4) for high-fidelity aerial scenes and AirSim for UAV control and sensing. Experiments run on Ubuntu 20.04 with an Intel Xeon Silver 4116 CPU and eight RTX 4090 GPUs (24 GB each). We evaluate our method on a public benchmark and our own dataset with explicit landmark annotations: **Public dataset:** OpenFly-AirSim contains 1,210 instruction-trajectory pairs from six AirSim scenes in OpenFly (Gao et al., 2026). The average instruction length, trajectory length, and number of stages are 67.1 words, 82.6 m, and 2.86, respectively. **Our dataset:** We select the four scenes with the highest simulation fidelity and visual quality to construct LandmarkFly, which contains 1,000 aligned instruction-trajectory-landmark samples. The corresponding averages are 94.2 words, 410.8 m, and 3.62 stages. For ablation studies, we also selected 20% of the data (i.e., 200 tasks) to serve as the LandmarkFly-mini dataset. More details on LandmarkFly are included in Appendix D.

Evaluation Metrics. **Standard metrics:** We report Success Rate (SR), Oracle Success Rate (OSR), Navigation Error (NE), and Success weighted by Path Length (SPL) to assess destination arrival and path efficiency, using a 20 m success threshold. **Soft metrics:** To soften the fixed distance threshold for large landmarks, Soft Success Rate (SSR) and Soft Oracle Success Rate (SOSR) assign full credit within 20 m and decrease linearly to zero at 30 m. **Stage-level evaluation metrics:** Conventional scores emphasize final outcomes rather than intermediate-stage performance. Landmark Visibility Rate (LVR), Grounding Point Accuracy (GPA), and Success Rate by Stage (SRS) assess initial visibility, grounding accuracy, and sequential progress, respectively; SRS applies SSR scoring to individual landmarks. **Safety and efficiency:** Robust Minimum Clearance (RMC) aggregates ground-excluded central-view depth by its per-step 0.1 quantile, stage mean, and task minimum. Mean Task Duration (MTD) includes parsing, grounding, and control time but excludes camera acquisition. More detailed explanations of the metrics are included in Appendix E.

Baselines and Implementation Details. We compare the task-trained OpenVLA-based OpenFly-Agent (Gao et al., 2026) with zero-shot See-Point-Fly (SPF) (Hu et al., 2025) and Fly0 (Xu et al., 2026b), which do not explicitly address viewpoint changes and grounding continuity across stages. SPF, Fly0, and AheadFly share **LocateAnything-3B** (Wang et al., 2026a) and **Qwen3-Max** for grounding and parsing; multi-stage parsing is supplied to the baselines. This focuses comparison on thinking and acting ahead to improve grounding continuity. As minimal baselines, we pair Landmark Description Extraction (LDE) or Transition-based Instruction Segmentation (TIS) with Directly Straight Forward (DSF) control toward the predicted target. More details are included in Appendix F.

Table 1: Overall results on OpenFly-AirSim (left) and LandmarkFly (right). OpenFly-Agent is trained on OpenFly and the remaining methods are zero-shot. NE is in meters and all other metrics are percentages. Bold and underlined values are the best and second-best results, respectively.

Method	OpenFly-AirSim						LandmarkFly								
	Standard metrics				Soft metrics		Standard metrics				Soft metrics		Stage-level metrics		
	SR↑	OSR↑	NE↓	SPL↑	SSR↑	SOSR↑	SR↑	OSR↑	NE↓	SPL↑	SSR↑	SOSR↑	LVR↑	GPA↑	SRS↑
OpenFly-Agent	22.64	52.23	<u>132.71</u>	20.40	26.17	57.70	1.90	6.90	334.35	1.66	2.23	8.32	53.96	–	41.63
SPF+TIS	<u>24.71</u>	52.89	160.95	<u>23.40</u>	<u>26.83</u>	58.09	10.80	17.00	275.94	10.55	11.78	18.62	45.41	57.68	58.66
Fly0+TIS	23.22	52.31	156.04	21.46	25.23	58.51	<u>37.90</u>	<u>51.10</u>	<u>129.36</u>	<u>33.75</u>	<u>41.34</u>	<u>54.29</u>	<u>63.76</u>	<u>80.25</u>	<u>81.32</u>
DSF+LDE	22.23	48.84	153.73	21.07	24.80	53.78	10.00	16.60	307.44	9.82	11.05	18.16	45.11	56.02	57.18
DSF+TIS	24.71	<u>54.63</u>	172.25	23.29	26.75	<u>59.15</u>	11.20	17.90	275.28	11.18	12.51	19.76	45.69	58.05	59.65
AheadFly (Ours)	**30.17**	**56.45**	**106.26**	**28.26**	**32.93**	**61.02**	**70.30**	**76.80**	**50.48**	**62.60**	**72.77**	**78.79**	**70.32**	**87.80**	**91.07**

4.2 Overall Results

Results on OpenFly-AirSim. As presented in the left part of Table 1, modular zero-shot methods generally match or exceed the task-specifically trained OpenFly-Agent, and DSF+TIS surpasses DSF+LDE, confirming TIS as the stronger parsing strategy. AheadFly leads on every metric, though its margin narrows to +1.82 and +1.87 points on OSR and SOSR. There, SPF+TIS and Fly0+TIS barely differ from the differently architected OpenFly-Agent and fail to surpass DSF+TIS, the minimal baseline that shares their TIS strategy. We attribute this to OpenFly-AirSim's shorter, straighter GT trajectories: most methods can reach or overshoot the endpoint in one stage of motion, which inflates the OSR-related metrics and lets the less maneuverable SPF+TIS beat Fly0+TIS on SR.

Results on LandmarkFly. As shown in the right part of Table 1, OpenFly-Agent collapses (all four success metrics below 10%), while every modular method, including the two minimal baselines, outperforms it. SPF+TIS's OpenFly-AirSim advantage does not carry over: it now trails Fly0+TIS on every metric. The stage-level metrics expose the cause: the baselines neglect viewpoint changes, so lower landmark visibility (LVR) breaks grounding continuity (GPA) and stage completion (SRS), causing early failure. Fig. 4 shows the same qualitatively: Fly0 grounds a non-instructed object after occlusion and deviates, whereas AheadFly stays stable. LandmarkFly's longer instructions and more stages further amplify these differences (per-difficulty results in Appendix G).

Table 2: Runtime efficiency and obstacle-avoidance (OA) performance. MTD (s) is split into parsing (Plan), grounding (Grd) and control (Act); RMC is the robust minimum clearance in meters.

Method	Runtime efficiency (MTD)				OA
	Plan↓	Grd↓	Act↓	All↓	RMC↑
OpenFly-Agent	–	–	–	29.89	23.59
SPF+TIS	3.86	<u>7.03</u>	49.59	60.48	33.88
Fly0+TIS	**3.51**	**7.02**	9.22	19.75	**47.07**
DSF+LDE	<u>3.68</u>	7.74	**2.85**	**14.27**	15.40
DSF+TIS	3.86	8.01	<u>3.02</u>	<u>14.89</u>	17.78
AheadFly(Ours)	5.80	7.24	3.38	16.42	<u>46.50</u>

Runtime Efficiency and Obstacle Avoidance. Fly0 and AheadFly, which ground landmarks and plan routes stage by stage, are substantially faster than SPF, which grounds and controls frame by frame, and the end-to-end OpenFly-Agent (Table 2). AheadFly is faster than Fly0 overall, with only the two DSF baselines being faster. Fly0 relies on LiDAR point clouds, depth maps, and camera extrinsics to obtain open flight corridors, whereas AheadFly uses depth maps alone to achieve comparable obstacle avoidance while also providing lookahead path adjustment and runtime efficiency.

4.3 Ablation Studies

Ablations of Core Mechanisms. Compared with TIS and LDE, TPG improves success rates as well as LVR, GPA, and SRS, but its gains are limited without viewpoint adjustment (Table 3). TIS consistently outperforms LDE, indicating that stage-organized prompts are better suited to multi-stage navigation than directly extracted landmark descriptions. Removing either viewpoint adjustment degrades performance, with pre-grounding viewpoint adjustment having a larger effect on landmark visibility. Combining both adjustments yields greater gains than either alone, demonstrating synergy between the acting-ahead mechanisms. Parsing and viewpoint adjustment are also synergistic, and the strong positive correlations of GPA with SR and OSR support improving final

Table 3: Ablations of the core mechanisms on LandmarkFly-mini (200 tasks). Variants A–K enable different task-parsing strategies (TPG/TIS/LDE), view adjustments (Pre: pre-grounding view adjustment; Look: lookahead path adjustment) and motion controllers (MFG/DSF); ✓ and × mark enabled and disabled components.

ID	Task parsing			View adjustment		Motion control		Standard metrics				Soft metrics		Stage-level metrics		
	TPG	TIS	LDE	Pre	Look	MFG	DSF	SR↑	OSR↑	NE↓	SPL↑	SSR↑	SOSR↑	LVR↑	GPA↑	SRS↑
A	✓	×	×	✓	✓	✓	×	**66.50**	**74.50**	**58.25**	**58.88**	**69.29**	**76.52**	**69.66**	**85.87**	**90.28**
B	×	✓	×	✓	✓	✓	×	42.00	54.00	127.05	36.74	44.33	56.66	64.76	80.78	81.59
C	×	×	✓	✓	✓	✓	×	35.50	48.00	147.41	31.94	38.36	49.93	62.16	76.97	76.08
D	✓	×	×	✓	×	✓	×	40.00	49.50	120.54	33.87	41.29	52.03	61.83	77.99	80.53
E	✓	×	×	×	✓	✓	×	18.50	24.50	251.96	18.57	19.69	26.64	47.01	59.26	60.78
F	✓	×	×	×	×	✓	×	18.50	23.00	279.48	17.56	19.23	24.23	45.10	55.81	58.68
G	×	✓	×	×	×	✓	×	15.00	19.50	284.26	13.20	14.40	19.74	44.94	55.70	57.09
H	×	×	✓	×	×	✓	×	10.00	15.00	292.18	10.59	11.60	18.58	44.29	54.58	55.69
I	✓	×	×	✓	×	×	✓	27.50	40.00	207.66	24.32	29.40	42.76	55.61	72.14	74.95
J	×	✓	×	×	×	×	✓	9.00	17.00	307.98	9.08	9.94	18.69	41.32	50.83	54.91
K	×	×	✓	×	×	×	✓	8.00	15.00	340.03	7.80	8.74	16.13	39.95	50.26	52.54

task performance through better intermediate grounding quality. A detailed analysis of the correlation is provided in Appendix H.

Module Substitutions. Without lookahead path adjustment, the MFG controller still beats the DSF controller (variants D and I in Table 3), likely because obstacle avoidance leaves the UAV in a more open position with a clearer view for the next stage. Replacing the grounding VLM with Qwen3-VL-Flash or Qwen3.8-Flash changes the metrics only marginally, with no backbone dominating (Table 4); the gains thus come from the thinking-and-acting-ahead mechanisms, not the VLM backbone.

Table 4: Effect of the grounding VLM backbone on LandmarkFly. Only the grounding backbone is replaced; the task parsing, view adjustment and motion control modules are unchanged.

Backbone	Standard metrics				Stage-level metrics		
	SR↑	OSR↑	NE↓	SPL↑	LVR↑	GPA↑	SRS↑
Qwen3.8-Flash	70.50	**78.50**	57.97	62.37	69.86	86.40	91.73
Qwen3-VL-Flash	**71.00**	77.00	52.79	**62.98**	70.09	87.12	**92.32**
LocateAnything	70.30	76.80	**50.48**	62.60	**70.32**	**87.80**	91.07

5 Limitations and Future Work

Our current framework assumes stage goals to be static landmarks whose positions are fixed before acting, and it has not been verified on moving targets. In addition, there is still room for improvement in fault tolerance: when errors occur in the preceding grounding, the guidance from TPG may further aggravate route deviation, and the framework currently lacks a mechanism to detect or recover from such a mismatch. On the dataset side, although LandmarkFly provides explicit landmark-centric annotations, the AVLN task that merely requires approaching landmarks in sequence still needs further study to cover richer forms of instruction. Future work will focus on periodic re-grounding for dynamic targets, per-stage completion verification with backtracking and re-planning, and predicting a set of waypoints at once to construct AVLN benchmarks with more diverse instructions and richer trajectory requirements.

6 Conclusion

We introduced AheadFly, a training-free framework for multi-stage aerial vision-and-language navigation that addresses grounding continuity under changing UAV viewpoints. AheadFly organizes instructions into landmark queries and transition cues, then combines pre-grounding view adjustment, lookahead path adjustment, and obstacle-aware motion control to prepare observations across stages. We also presented LandmarkFly and stage-level metrics to evaluate landmark visibility, grounding accuracy, and ordered navigation progress. Results on OpenFly-AirSim and LandmarkFly demonstrate improved navigation and stage-level grounding, highlighting the importance of coordinating what to look for, where to look, and where to look from.

AI USE STATEMENT

In this work, we used generative AI tools to generate and reformat the synthetic LandmarkFly dataset, and to assist in implementing the task-parsing, grounding, and motion-control components of AheadFly. We have not used generative AI tools to propose or refine hypotheses, to design the experimental methodology, to interpret the experimental results, or to formulate mathematical claims, and proof-writing is not applicable to this work. Additionally, we used generative AI tools to edit the manuscript for readability, to check reference formatting, and to translate the prompt templates reproduced in Appendix I and Appendix J. We have reviewed all AI-assisted work and we take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

All experiments in this work were conducted in simulation, using the Unreal Engine 4 and AirSim environments described in Section 4.1. The LandmarkFly dataset was generated entirely from rendered scenes by the automated, landmark-centric pipeline in Section 3.5; it contains no human subjects, no human demonstrations, and no personal or identifiable information. The dataset and the prompt templates will be released for research use. We note that autonomous aerial navigation is a dual-use capability: while the applications motivating this work are inspection, emergency response, and urban management, the same techniques could be applied to surveillance or to navigation in areas where flight is restricted. Our contribution is a navigation method evaluated in simulation with publicly available benchmarks, and we will release our artifacts to support transparent follow-up work on the safety and robustness of aerial navigation upon publication.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we describe the three components of AheadFly in Section 3.2 (LLM-based structured task parsing and Task Planning Graph construction), Section 3.3 (view-aware landmark grounding and 3D waypoint generation) and Section 3.4 (multi-force-guided motion control with pre-grounding view adjustment and lookahead path adjustment), with the full derivations in Appendices A–C. Section 3.5 presents the landmark-centric data generation pipeline and the annotation procedure for LandmarkFly, whose composition and statistics are detailed in Appendix D. The simulation environment, hardware, benchmarks, evaluation metrics and baselines are specified in Section 4.1; complete metric definitions and baseline settings are given in Appendices E and F, and the supplementary analyses (results grouped by task difficulty, and the grounding–success correlation) in Appendices G and H. All prompts for instruction parsing and data generation are reproduced verbatim in Appendices I and J. Code and the LandmarkFly dataset will be released upon publication.

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 *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 3674–3683. IEEE, 2018.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. Speaker-follower models for vision-and-language navigation. *Advances in neural information processing systems*, 31, 2018.

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:2410.08500*, 2024.

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.

Weituo Hao, Chunyuan Li, Xiujun Li, Lawrence Carin, and Jianfeng Gao. Towards learning a generic agent for vision-and-language navigation via pre-training. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13134–13143. IEEE, 2020.

Zongtao He, Liuyi Wang, Shu Li, Qingqing Yan, Chengju Liu, and Qijun Chen. A multilevel attention network with sub-instructions for continuous vision-and-language navigation: Z. he et al. *Applied Intelligence*, 55(7):657, 2025.

Yicong Hong, Cristian Rodriguez, Qi Wu, and Stephen Gould. Sub-instruction aware vision-and-language navigation. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 3360–3376, 2020.

Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. Vln bert: A recurrent vision-and-language bert for navigation. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pp. 1643–1653, 2021.

Chih Yao Hu, Yang-Sen Lin, Yuna Lee, Chih-Hai Su, Jie-Ying Lee, Shr-Ruei Tsai, Chin-Yang Lin, Kuan-Wen Chen, Tsung-Wei Ke, and Yu-Lun Liu. See, point, fly: A learning-free vlm framework for universal unmanned aerial navigation. In *Conference on Robot Learning*, pp. 4697–4708. PMLR, 2025.

Wen Jiang, Li Wang, Kangyao Huang, Wei Fan, Jinyuan Liu, Shaoyu Liu, Hongwei Duan, Bin Xu, and Xiangyang Ji. Longfly: Long-horizon uav vision-and-language navigation with spatiotemporal context integration. *arXiv preprint arXiv:2512.22010*, 2025.

Wen Jiang, Kangyao Huang, Li Wang, Wang Xu, Wei Fan, Jinyuan Liu, Shaoyu Liu, Hanfang Liang, Hongwei Duan, Bin Xu, et al. Spatialfly: Geometry-guided representation alignment for uav vision-and-language navigation in urban environments. *arXiv preprint arXiv:2603.21046*, 2026.

Tianshun Li, Tianyi Huai, Zhen Li, Yichun Gao, Haoang Li, and Xinhu Zheng. Skyvln: Vision-and-language navigation and nmpc control for uavs in urban environments. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 17199–17206. IEEE, 2025.

Yijian Li, Changze Li, Hantian Shi, Jiaying Luo, Jiyuan Cai, Ming Yang, and Tong Qin. Agenticnav: Zero-shot vision-and-language navigation as a tool-calling harness. *arXiv preprint arXiv:2606.10577*, 2026.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International conference on robotics and automation (ICRA)*, pp. 9493–9500. IEEE, 2023.

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, 2023.

Kailin Lyu, Kangyi Wu, Pengna Li, Wenxuan Song, Di Wu, Jianwei He, Junting Chen, Ning Yang, Zebin Han, Kaiwen Luo, et al. From instruction following to cognitive navigation: A survey on the evolution of vision-and-language navigation. Technical report, CSPaper OpenPrint, 2026.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 32441–32450, June 2026.

Zhe Peng, Guohua Wu, Biao Luo, and Ling Wang. Multi-uav cooperative pursuit strategy with limited visual field in urban airspace: A multi-agent reinforcement learning approach. *IEEE/CAA Journal of Automatica Sinica*, 12(7):1350–1367, 2025.

Liang Qin, Min Wang, Xingyu Lu, Aowen Qiu, Wengang Zhou, and Houqiang Li. Uast: Unified active search and tracking for arbitrary targets with uavs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13464–13473, 2026.

Oleg Sautenkov, Yasheerah Yaqoot, Artem Lykov, Muhammad Ahsan Mustafa, Grik Tadevosyan, Aibek Akhmetkazy, Miguel Altamirano Cabrera, Mikhail Martynov, Sausar Karaf, and Dzmitry Tsetserukou. Uav-vla: Vision-language-action system for large scale aerial mission generation. In *2025 20th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 1588–1592. IEEE, 2025.

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

Hongyu Song, Rishabh Dev Yadav, Cheng Guo, and Wei Pan. Soranav: Adaptive uav task-centric navigation via zeroshot vlm reasoning. *arXiv preprint arXiv:2510.25191*, 2025.

Shihao Wang, Shilong Liu, Yuanguo Kuang, Xinyu Wei, Yangzhou Liu, Zhiqi Li, Yunze Man, Guo Chen, Andrew Tao, Guilin Liu, et al. Locateanything: Fast and high-quality vision-language grounding with parallel box decoding. In *European Conference on Computer Vision*, pp. 336–357. Springer, 2026a.

Teng Wang, Yanting Lu, and Ruize Wang. Autotraces: Autoregressive trajectory forecasting via multimodal large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4054–4064, June 2026b.

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.

Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6622–6631. IEEE, 2019.

Ruipu Wu, Yige Zhang, Jinyu Chen, Linjiang Huang, Shifeng Zhang, Xu Zhou, Liang Wang, and Si Liu. Aeroduo: Aerial duo for uav-based vision and language navigation. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 2576–2585, 2025.

Zihao Xin, Wentong Li, Yixuan Jiang, Ziyuan Huang, Bin Wang, Piji Li, Jianke Zhu, Jie Qin, and Shengjun Huang. Agentvln: Towards agentic vision-and-language navigation. In *European Conference on Computer Vision*, pp. 114–132. Springer, 2026.

Peng Xu, Zhengnan Deng, Jiayan Deng, Zonghua Gu, and Shaohua Wan. Aerovla: A vision-language-action model for uav navigation via minimalist end-to-end control. In *European Conference on Computer Vision*, pp. 289–305. Springer, 2026a.

Zhenxing Xu, Brikit Lu, Weidong Bao, Zhengqiu Zhu, Junsong Zhang, Hui Yan, Wenhao Lu, and Ji Wang. Fly0: Decoupling semantic grounding from geometric planning for zero-shot aerial navigation. *arXiv preprint arXiv:2602.15875*, 2026b.

Fanglong Yao, Youzhi Liu, Wenyi Zhang, Zhengqiu Zhu, Chenglong Li, Nayu Liu, Peng Hu, Yuanchang Yue, Kaiwen Wei, Xin He, et al. Aeroverse-review: Comprehensive survey on aerial embodied vision-and-language navigation. *The Innovation Informatics*, 1(1):100015, 2025.

Lu Yue, Yue Fan, Shiwei Lian, Yu Zhao, Jiaxin Yu, Liang Xie, and Feitian Zhang. Spatial-vln: Zero-shot vision-and-language navigation with explicit spatial perception and exploration. *arXiv preprint arXiv:2601.12766*, 2026.

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

Xinyuan Zhang, Yonglin Tian, Fei Lin, Yue Liu, Jing Ma, Xiao Wang, Kornélia Sára Szatmáry, and Fei-Yue Wang. Logisticsvln: Vision-language navigation for low-altitude terminal delivery based on agentic uavs. In *2025 IEEE 28th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 4437–4442. IEEE, 2025b.

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

Yihao Zhao, Enze Zhu, Zhan Chen, Benkui Zhang, Wenxiang Huo, Xinyu Zhao, and Ying Chang. Embodied ai in the sky: A comparative review of uav embodied ai, from autonomous remote sensing to task execution. *Remote Sensing*, 18(10):1509, 2026.

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.

Wanrong Zheng, Yunhao Ge, and Laurent Itti. Three-step nav: A hierarchical global–local planner for zero-shot vision-and-language navigation. In Emtiyaz Khan, Yingzhen Li, Arno Solin, and Aaditya Ramdas (eds.), *Proceedings of The 29th International Conference on Artificial Intelligence and Statistics*, volume 300 of *Proceedings of Machine Learning Research*, pp. 4645–4653. PMLR, 02–05 May 2026b. URL https://proceedings.mlr.press/v300/zheng26b.html.

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

Jiaxu Zhou, Shaobo Wang, Zhiyuan Yang, Zhenjun Yu, and Tao Li. Freefly-thinking: Aligning chain-of-thought reasoning with continuous uav navigation. *arXiv preprint arXiv:2603.07181*, 2026.

Xin Zhou, Zhepei Wang, Hongkai Ye, Chao Xu, and Fei Gao. Ego-planner: An esdf-free gradient-based local planner for quadrotors. *IEEE Robotics and Automation Letters*, 6(2):478–485, 2020.

APPENDIX

This appendix provides supplementary details and analyses that complement the main paper. Appendix A illustrates the construction of the Task Planning Graph (TPG) with a concrete navigation instruction, showing how the task parsing module Φ_{Plan} divides an instruction into stages and resolves each stage into an explicit landmark query and a transition direction, and Appendix B details how the repulsive vector of the motion controller Φ_{Act} is computed from global and local depth gradients. Appendix C describes how the magnitudes of the bias, attractive and repulsive force vectors are balanced through the stage-progress and corridor-openness factors, and Appendix D documents the specifications of the LandmarkFly benchmark, covering its scene selection, trajectory and landmark statistics, and difficulty tiers. Appendix E defines the evaluation metrics adopted in this work, including the standard and soft success measures as well as the stage-level and safety/efficiency metrics, and Appendix F summarises the baseline settings used for a fair comparison. Appendix G reports the results grouped by task difficulty, while Appendix H examines the association between grounding continuity and task success. Finally, Appendices I and J reproduce the prompt contents used by the method and by the data-generation pipeline, respectively.

A CONSTRUCTING THE TASK PLANNING GRAPH

This section illustrates, with a concrete example, how AheadFly's task parsing module Φ_{Plan} constructs a structured **Task Planning Graph (TPG)** from a language instruction, as shown in Fig. 5.

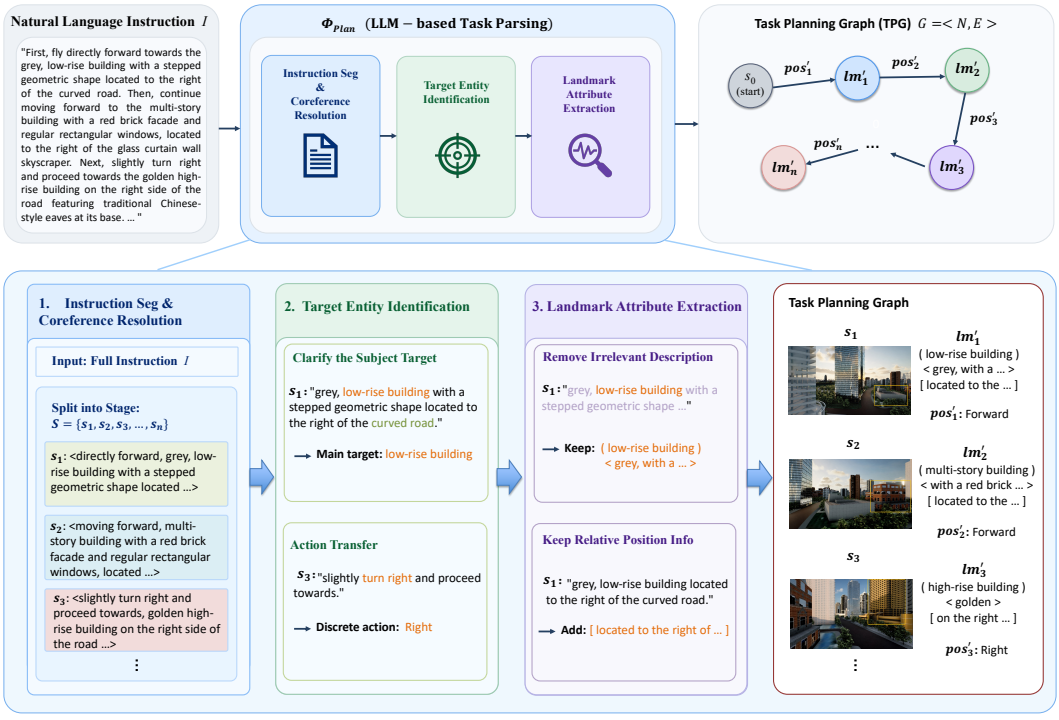

Figure 5: Task Planning Graph (TPG) construction. Φ_{Plan} applies instruction segmentation with coreference resolution, target entity identification and landmark attribute extraction, so that spatial relations enter the node and the discrete action enters the edge: nodes lm'_i hold a landmark's category, attributes $\langle\cdot\rangle$ and relative position $[\cdot]$, and edges pos'_i hold the transition direction.

Before each navigation task, Φ_{Plan} receives the complete natural-language instruction I and analyzes it to divide the task into stages $S = \{s_1, s_2, ..., s_M\}$. A stage $s_i =< pos_i, lm_i >$ indicates that a UAV near landmark lm_{i-1} (lm_0 denotes the task start) should attend to relative direction pos_i and approach landmark lm_i. Φ_{Plan} then structures pos_i into a concise direction, such as "left," "right," or "ahead," and clarifies the description of lm_i through the following operations:

(1) **Coreference resolution:** Resolve references such as "that building," "the previous building," "a building with an appearance similar to the previous landmark," and "it" using the instruction context, so that each stage description is sufficiently clear and independently identifiable;

(2) **Target entity identification:** Analyze the current description to determine the key target that the UAV is actually required to approach, and distinguish the subject from the object in the instruction. This prevents the UAV from incorrectly focusing on a description of a non-landmark object. For example, in "go to B located to the left of A," B is the target;

(3) **Landmark Attribute Extraction:** Remove content unrelated to the landmark's visual features while retaining its relative position with respect to other objects. This relation makes the landmark more identifiable, so it is appended to the target description as part of the landmark's features, e.g., extracting "B, located to the left of A" from "go to B located to the left of A."

Finally, Φ_{Plan} combines the two types of parsed information to construct a text-based directed graph $G =< N, E >$ as the TPG. Here, $N = \{lm'_1, lm'_2, ..., lm'_M\}$ and $E = \{pos'_1, pos'_2, ..., pos'_M\}$ denote the sets of **explicit landmark queries** and **inter-stage transition directions**, respectively.

B GENERATING THE REPULSIVE VECTOR FROM DEPTH GRADIENTS

The detailed process for computing the **repulsive vector** V_{rep} in the **Multi-Force-Guided Motion Controller** Φ_{Act} is shown in Fig. 6.

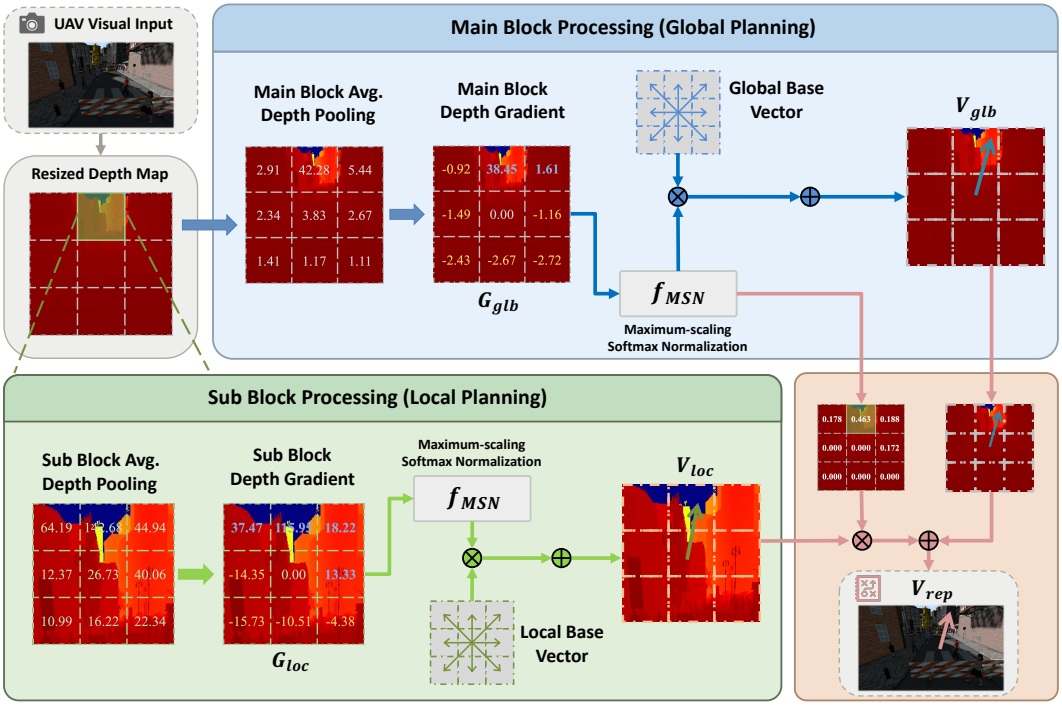

Figure 6: Repulsive-vector generation. The depth map is average-pooled and differentiated over nine main blocks (global planning) and nine sub-blocks per block (local planning); maximum-scaling softmax converts the positive gradients into directional weights, which are combined with eight basis directions into V_{glb} and V_{loc} and fused into V_{rep}.

For scene depth information $D \in R^{(9H)\times(9W)}$, where larger values indicate greater distance, we first scale the values to 0–255 and divide the map into nine equal main blocks $D^i_{blk} \in R^{(3H)\times(3W)}, 1 \leq i \leq 9$. Each main block is further divided into nine sub-blocks in the same way, yielding 81 sub-blocks in total:

$$D^i_{blk} = \{d^{i1}_{blk}, d^{i2}_{blk}, ..., d^{i9}_{blk}\}, d^{ij}_{blk} \in R^{H\times W}, 1 \leq i \leq 9. \tag{3}$$

For each main block and sub-block, we use average pooling $Mean(\cdot)$ to compute the mean depth value of its region:

$$\begin{cases} \overline{D^i_{blk}} = Mean(D^i_{blk}) = \frac{1}{(3H)\times(3W)} \sum_{x \in D^i_{blk}} x \\ \overline{d^{ij}_{blk}} = Mean(d^{ij}_{blk}) = \frac{1}{H \times W} \sum_{x \in d^{ij}_{blk}} x \end{cases} . \tag{4}$$

For the full depth map D, we then compute the mean-depth gradients of the eight surrounding main blocks relative to the center block (the main block with $i = 5$), obtaining eight global depth gradients. For a main block D^i_{blk}, the same procedure yields eight local depth gradients within that block:

$$\begin{cases} G_{glb} = \{\overline{D^i_{blk}} - \overline{D^5_{blk}}\}_{1 \le i \le 9, i \ne 5} \\ G^i_{loc} = \{\overline{d^{ij}_{blk}} - \overline{d^{i5}_{blk}}\}_{1 \le j \le 9, j \ne 5} \end{cases} . \tag{5}$$

These eight gradients directly indicate the tendency to shift from the center block toward each surrounding block to avoid nearby obstacles, i.e., the strength with which the UAV heading should turn toward that location. A negative value indicates a tendency in the opposite direction. We therefore negate every negative gradient and transfer it to the gradient of the opposite block, obtaining non-negative depth gradients ${G_{glb}}'$ and ${G^i_{loc}}'$. We then apply maximum-scaling Softmax normalization $MSN(\cdot)$ to the positive gradients, producing weights that sum to one while keeping their numerical differences within a controlled range. For the global gradients:

$$W_{glb} = MSN({G_{glb}}') = Softmax(MaxScaling({G_{glb}}')) = \{w_i\}_{1 \le i \le 9, i \ne 5}, \tag{6}$$

where $MaxScaling(\cdot)$ and $Softmax(\cdot)$ denote scaling each value to 0–1 by the maximum value and Softmax normalization, respectively. Let W^i_{loc} denote the result obtained from the local gradients. We next define eight reference direction vectors whose components are 0 or 1 and compute their weighted "resultant force" to obtain the global and local obstacle-avoidance planning vectors V_{glb} and $V^i_{loc}, 1 \le i \le 9$, respectively. The local vectors are then combined into the global vector according to the weights of their corresponding main blocks, providing a finer adjustment to the direction and magnitude of V_{glb}. The resulting obstacle-avoidance planning vector V_{rep} intuitively reflects the direction and strength of the avoidance action recommended by the controller:

$$V_{rep} = \alpha \cdot V_{glb} + \beta \cdot \sum_{1 \le i \le 9, i \ne 5} w_i \cdot V^i_{loc} + \gamma \cdot V^5_{loc}, \tag{7}$$

where α, β, and γ are scaling parameters selected to adjust the numerical scale and ensure that every component of the final obstacle-avoidance planning vector remains in the range from -1 to 1. Its direction and magnitude intuitively reflect the direction and strength of the avoidance action recommended by the controller. Because V_{rep} is computed according to the obstacle-avoidance principle, it is effectively **a "repulsive force" exerted by scene depth information on the UAV's forward direction**.

C Controlling Force-Vector Magnitudes

In the motion controller Φ_{Act}, the force vectors V_{bias}, V_{att}, and V_{rep} are weighted and combined to compute the current-frame **action vector** V_{act}, as shown in Equation. 2. The factors η, δ used to control the contribution of each term and the output scale are determined by:

$$\eta = \begin{cases} \frac{|V_{att}|}{|V_{att}|+|V_{rep}|}, & \text{if } 0.3 \le p \le 0.7 \\ max(p, 1-p), & \text{else} \end{cases}, \tag{8}$$

$$\delta = \frac{50}{50 + \mathrm{Q}_{0.2}(D^5_{blk})}, \tag{9}$$

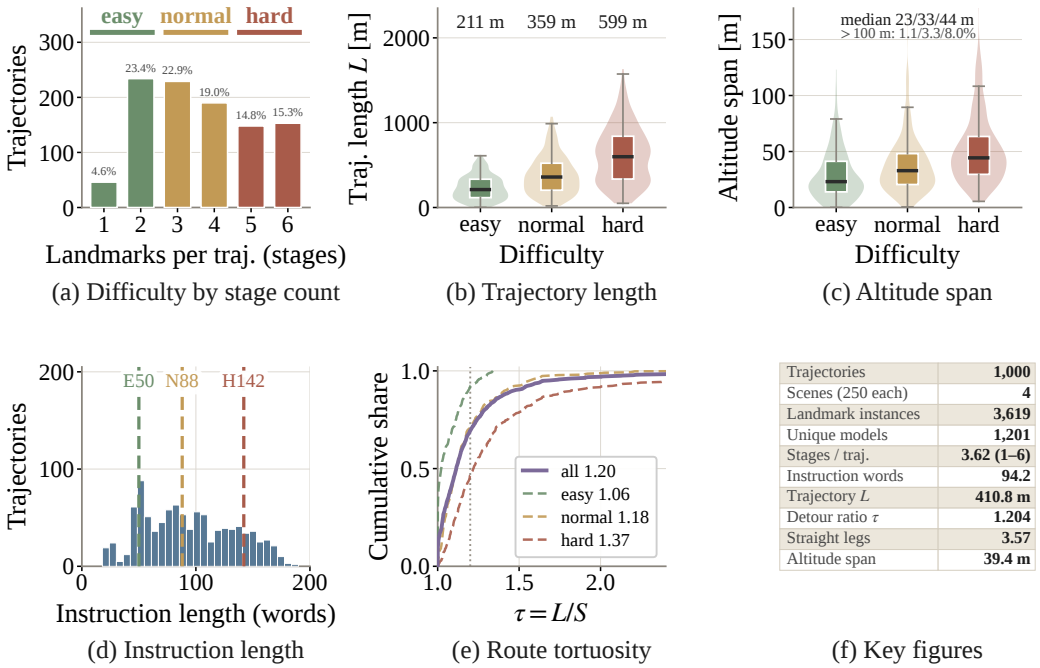

Figure 7: Specifications of LandmarkFly (1,000 trajectories, four scenes). (a) Landmarks (stages) per trajectory by difficulty tier (easy 1–2, normal 3–4, hard 5–6); (b) trajectory length and (c) altitude span per tier; (d) instruction length; (e) cumulative distribution of the detour ratio $\tau = L/S$; (f) key statistics (3.62 stages and 410.8 m per trajectory). All statistics increase monotonically across tiers.

where p denotes the normalized distance between the UAV and the grounded landmark, obtained from depth information. It indicates the completion progress of the stage-level subtask as a continuous value: 1 means that the current stage has just begun, and 0 means that it has finished. $Q_{0.2}(D^5_{blk})$ denotes the 0.2 quantile of the central main block in the depth map. All constants are selected based on experience and best practices.

Finally, η adjusts the influence of the attractive vector V_{att} and the bias vector V_{bias} according to vector magnitude and flight stage, allowing Φ_{Act} to **adapt its actions to the relative need for direct flight and obstacle avoidance**. δ attenuates repulsion according to the openness of the forward corridor, **reducing yaw oscillations** caused by depth variations of low-impact objects outside the flight corridor and stabilizing the trajectory.

D Dataset Specifications

The detailed specifications of our LandmarkFly benchmark across various aspects are illustrated in Fig. 7.

Dataset size and composition. LandmarkFly contains 1,000 AVLN trajectories evenly distributed over four simulated scenes (250 trajectories each), which were selected from the six available AirSim scenes in OpenFly (Gao et al., 2026) for their highest simulation fidelity and visual quality, as shown in Fig. 8. Each trajectory consists of a number of stages, in which the UAV approaches the landmark objects described in the instruction in turn. The benchmark comprises 3,619 landmark instances drawn from 1,201 distinct 3D landmark models, and landmarks are almost never reused across scenes, which guarantees the diversity of navigation targets. A trajectory contains 3.62 stages on average (range 1–6); based on this we group the tasks into three difficulty tiers — easy (1–2 stages), normal (3–4) and hard (5–6) — containing 280, 419 and 301 trajectories respectively.

Instructions and trajectory geometry. Instructions are 94.2 words long on average (up to 190 words) with a vocabulary of 1,271, and every instruction is unique, which rules out evaluation leakage through templated text. Trajectory geometry is measured under a single consistent protocol: the polyline is first projected onto the top-down (XY) plane and consecutive duplicate nodes are



(a) env_airsim_16 (b) env_airsim_18

(c) env_airsim_26 (d) env_airsim_sh

Figure 8: The four AirSim scenes used in LandmarkFly: (a) env_airsim_16, (b) env_airsim_18, (c) env_airsim_26 and (d) env_airsim_sh, selected from the six AirSim scenes in OpenFly (Gao et al., 2026) for fidelity and visual quality; each contributes 250 of the 1,000 trajectories.

removed (threshold 0.01 m); we then define the path length L, the start-to-goal straight-line span S and the detour ratio $\tau = L/S$. The mean trajectory length is 410.8 m and the mean detour ratio is 1.204, i.e. trajectories are on average 20% longer than the straight line; 30.7% of trajectories have $\tau > 1.2$ and 9.6% have $\tau > 1.5$, far above what straight-segment navigation benchmarks exhibit. After merging collinear nodes with a 0.1° tolerance, a trajectory consists of 3.57 straight legs on average, closely matching the mean number of stages (3.62); this indicates that every stage node corresponds to a genuine turning point rather than a redundant intermediate sample. The mean altitude span is 39.4 m, reflecting the vertical manoeuvring required by low-altitude tasks.

Validity of the difficulty tiers. The three tiers increase monotonically across all reported statistics: stages $2 \rightarrow 3 \rightarrow 6$, instruction length $50 \rightarrow 88 \rightarrow 142$ words, trajectory length $211 \rightarrow 359 \rightarrow 599$ m, detour ratio $1.023 \rightarrow 1.114 \rightarrow 1.219$ and straight legs $2 \rightarrow 3 \rightarrow 5$. This simultaneous increase along several independent axes indicates that difficulty defined by the stage count reflects a genuine difference in task complexity rather than an arbitrary cut on a single metric.

E Evaluation Metric Definitions

The commonly used standard metrics for AVLN tasks include the following AerialVLN (Liu et al., 2023):

- **Success Rate (SR):** A task is considered successful if and only if the UAV's final stopping position lies within the target neighborhood (20 m); SR is the proportion of successful tasks.
- **Oracle Success Rate (OSR):** A task is considered successfully visited if the UAV passes through the target neighborhood at any point during navigation; OSR is the proportion of tasks successfully visited.
- **Navigation Error (NE):** The distance between the UAV's final stopping position and the target location;
- **Success weighted by Path Length (SPL):** The product of the ratio between the ground-truth path length and the actual flight-path length and the success rate, measuring route efficiency and path similarity.

However, mainstream success metrics often judge arrival using only a **fixed distance threshold** (e.g., 20 m) and return a binary result of 0 or 1. In open low-altitude scenes, target objects are typically medium-sized or large buildings, so evaluations may differ substantially depending on where on the same building the UAV arrives. Simply enlarging the threshold (e.g., to 30 m) would

make the metric overly permissive and less informative. We therefore introduce **soft metrics** with a linearly decaying buffer interval, extending evaluation from whether the UAV reached the target location to how close it came:

- **Soft Success Rate (SSR):** An improved success metric whose value decreases linearly from 1 to 0 as NE increases from 20 m to 30 m;
- **Soft Oracle Success Rate (SOSR):** An improved oracle success metric that applies the same calculation as SSR to OSR;

In addition, existing metrics **emphasize final outcomes rather than intermediate-stage performance**. Although OSR and SPL partly account for the intermediate flight process, they are too coarse, partly because existing datasets lack explicit landmark annotations. Based on LandmarkFly, we introduce the following **stage-level evaluation metrics** to comprehensively evaluate AVLN baselines from the perspectives of landmark visibility at stage onset, consecutive grounding accuracy, and whether the UAV follows the instruction:

- **Landmark Visibility Rate (LVR):** The visibility of each key landmark before the beginning of a sub-stage, computed from the normalized Euclidean distance between the object's geometric center and the image center;
- **Grounding Point Accuracy (GPA):** The normalized Euclidean distance between the center of the target grounding box and the projection of the landmark's geometric center onto the image plane; its average is used as the grounding accuracy for one task;
- **Success Rate by Stage (SRS):** The proximity of the UAV to each stage landmark during flight, evaluated using the SSR calculation to represent the UAV's ability to follow the language instruction and complete navigation.

Finally, we provide two metrics for evaluating **obstacle-avoidance safety** and **mean module runtime**, reflecting baseline execution efficiency:

- **Robust Minimum Clearance (RMC):** At each step, the center depth of the observation is extracted after excluding the ground, and its 0.1 quantile is used as a robust proximity measure. The values are averaged within each stage to obtain the clearance; task-level RMC is the minimum across all stages, representing the worst-case safety margin during navigation;
- **Mean Task Duration (MTD):** The average task completion time excluding camera data acquisition. For modular architectures, this includes task parsing, landmark grounding, and motion control.

F Baseline Settings

We leave the original end-to-end OpenFly-Agent architecture unchanged. For fair comparison among modular AVLN architectures, the grounding modules of See-Point-Fly (Hu et al., 2025) and Fly0 (Xu et al., 2026b) use the same VLM as AheadFly, namely NVIDIA's LocateAnything-3B (Wang et al., 2026a), a model specialized for grounding. Although See-Point-Fly and Fly0 describe a similar stage-wise design, the open-source implementations of these methods lack dedicated parsing for multi-stage instructions. We therefore also supply Qwen3-Max-based instruction parsing and the required stage-level instructions, choosing TIS to match their design. This focuses the analysis on **the effectiveness of thinking and acting ahead in adapting to the dynamic characteristics of navigation tasks and improving grounding continuity**. We additionally introduce two minimal baselines, DSF+LDE and DSF+TIS, which use the LDE or TIS parsing strategy and a Directly Straight Forward (DSF) controller as basic zero-shot modular baselines. The DSF controller flies directly toward the target location predicted by the VLM.

G Results Grouped by Task Difficulty

We evaluate zero-shot methods on LandmarkFly, whose stages are explicitly defined and aligned with its landmarks, under different AVLN task difficulties. LandmarkFly contains at most 6 stages

Table 5: Performance of the zero-shot methods stratified by task difficulty on LandmarkFly. Tasks are grouped by the number of landmark stages: easy (1–2; $N = 280$), normal (3–4; $N = 419$) and hard (5–6; $N = 301$). NE is in meters and all other metrics are percentages. Bold and underlined values are the best and second-best results.

Difficulty	N	Method	Standard metrics				Soft metrics		Stage-level metrics		
			SR↑	OSR↑	NE↓	SPL↑	SSR↑	SOSR↑	LVR↑	GPA↑	SRS↑
Easy	280	SPF+TIS	27.14	35.36	142.87	25.76	29.05	36.78	50.26	65.05	73.68
		Fly0+TIS	<u>58.21</u>	<u>68.93</u>	<u>61.68</u>	<u>53.91</u>	<u>62.23</u>	<u>71.50</u>	<u>63.83</u>	<u>83.66</u>	<u>89.02</u>
		DSF+LDE	23.57	33.57	160.28	22.57	25.87	36.07	51.82	65.69	72.79
		DSF+TIS	26.43	35.00	133.77	25.57	29.01	37.98	52.41	67.69	74.81
		AheadFly(Ours)	**83.21**	**88.21**	**27.59**	**77.48**	**85.52**	**89.36**	**67.89**	**90.49**	**95.49**
Normal	419	SPF+TIS	6.92	13.60	254.40	7.14	7.72	15.62	47.03	58.86	59.81
		Fly0+TIS	<u>38.19</u>	<u>53.70</u>	<u>105.65</u>	<u>33.42</u>	<u>41.73</u>	<u>57.37</u>	<u>64.39</u>	<u>80.58</u>	<u>84.21</u>
		DSF+LDE	7.16	14.08	291.49	7.47	8.12	15.75	46.75	57.17	58.51
		DSF+TIS	7.88	15.99	253.10	8.36	9.15	17.88	46.94	58.94	60.98
		AheadFly(Ours)	**71.36**	**78.28**	**46.28**	**64.39**	**73.94**	**80.39**	**70.67**	**86.79**	**91.68**
Hard	301	SPF+TIS	1.00	4.65	429.70	1.15	1.36	5.91	38.63	49.16	43.10
		Fly0+TIS	<u>18.60</u>	<u>30.90</u>	<u>225.31</u>	<u>15.47</u>	<u>21.36</u>	<u>34.01</u>	<u>62.83</u>	<u>76.63</u>	<u>70.14</u>
		DSF+LDE	1.33	4.32	466.53	1.23	1.33	4.86	36.61	45.43	40.81
		DSF+TIS	1.66	4.65	437.81	1.71	1.82	5.43	37.71	47.85	43.71
		AheadFly(Ours)	**56.81**	**64.12**	**77.62**	**46.25**	**59.28**	**66.72**	**72.09**	**86.71**	**86.12**

per task. We classify tasks with one or two stages as "easy," those with three or four stages as "normal," and those with at least five stages as "hard," as mentioned in Appendix D. More stages generally imply **longer flight distances and longer, more complex language instructions**, placing greater demands on zero-shot baselines to process and integrate multimodal information and plan flight routes appropriately. We therefore consider it reasonable to associate stage count with task difficulty. The requirements for grounding continuity in multi-stage tasks and the effects of viewpoint dynamics become increasingly pronounced as task difficulty rises, making AheadFly's advantages in addressing these issues easier to observe. The grouped results are shown in table 5.

AheadFly leads on AVLN performance at every difficulty level, and the gaps in SR and SSR over the second-best method grow with task difficulty. This demonstrates strong robustness to the number of stages in multi-stage navigation and the effectiveness of the proposed design for improving multi-stage task success. Stage-level results further suggest that AheadFly's "thinking ahead" and "acting ahead" mechanisms play an important role in increasing landmark visibility (**higher LVR on harder tasks**), improving consecutive grounding accuracy (**maintaining GPA across difficulty levels**), and advancing multi-stage task completion (**a larger SRS gap over the second-best method**).

Overall, AheadFly brings clear gains on multi-stage AVLN tasks, and its advantage further increases as the number of stages grows and the tasks become more complex. This increase in difficulty exposes AVLN methods that are unable, or were not designed, to handle the effects of navigation-task dynamics.

H Correlation between Grounding Continuity and Task Success

As shown in Fig. 9, we examine the association between GPA, which evaluates continuous grounding accuracy in AVLN, and five navigation-success metrics (SR, SSR, OSR, SOSR, and SRS). Each observation is a *method–scene aggregate*: the mean score of one zero-shot baseline in one of the four LandmarkFly scenes, which averages over roughly 250 evaluation tasks. Five methods across four scenes give $n = 20$ observations. Two properties of these data constrain how the results should be read. First, the observations are not mutually independent: a method contributes to four observations and a scene contributes to five, so method and scene effects are shared across points and the

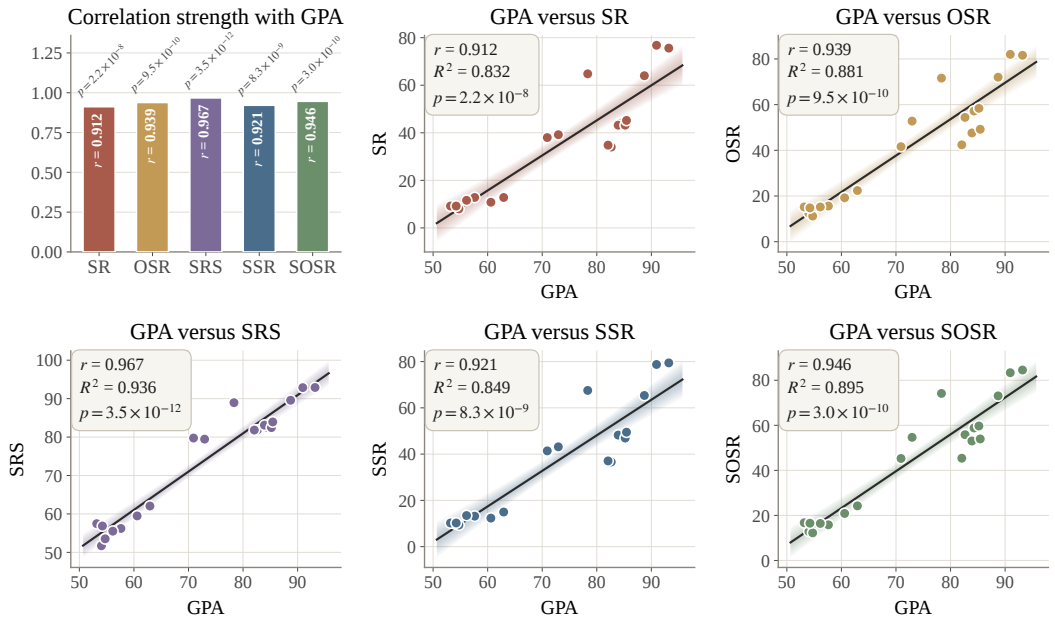

Figure 9: Association between grounding quality and navigation success. Left: Pearson correlation of GPA with SR, OSR, SRS, SSR and SOSR. Right: the corresponding method–scene aggregates ($n = 20$), annotated with r, R^2 and p. GPA is strongly associated with every success metric; the aggregates are not independent, so the analysis is descriptive.

effective degrees of freedom are far smaller than n. Second, the four scenes were selected for simulation fidelity rather than sampled at random, which restricts cross-scene variance. We therefore treat the following analysis as descriptive and do not attach calibrated significance levels to it.

Pearson correlation shows strong positive associations between GPA and each of SSR, SR, SOSR, OSR, and SRS ($r = 0.912 \sim 0.967$), with the strongest association for SRS ($r = 0.967$, $R^2 = 0.936$). Because the variables deviate from normality under the Shapiro–Wilk test, we also compute Spearman rank correlations as a robustness check and obtain consistent results ($\rho = 0.890 \sim 0.929$). To check that the trend is not produced by the shared method or scene structure, we additionally examine the association within each scene separately and, following Appendix G, within each difficulty stratum; the trend persists in both cases, although each of these subsets contains only five methods and is correspondingly weaker evidence. We note that GPA lies between 80 and 88 for most modular methods, which compresses the predictor range through a ceiling effect.

Taken together, these results indicate a stable and highly consistent positive association between GPA and navigation-success metrics: methods that ground intermediate landmarks more accurately also tend to complete multi-stage tasks more often. This is consistent with – but does not establish – a causal role for grounding quality in task success, since success and grounding share both trajectory outcomes and evaluation labels, and an association across method–scene aggregates cannot separate the contribution of a specific mechanism from the overall quality of the method that produced it. The controlled evidence for our design is the ablation in Table 3, where the two viewpoint adjustments are removed independently while all other components are held fixed. We therefore read this analysis as descriptive support for treating landmark grounding as a central component of modular AVLN architectures, and not as independent validation of AheadFly's mechanisms.

I Prompt Contents for the Method

Instruction parsing uses **Qwen3-Max at temperature 0.3**, and all prompts are used verbatim.

I.1 Instruction parsing (Φ_{Plan})

Guided by the prompt below, the task parsing module Φ_{Plan} parses instructions using either **Landmark Description Extraction (LDE)** or **Transition-based Instruction Segmentation (TIS)**.

Prompt 1 — Instruction parsing (Φ_{Plan})

Role Definition
Assume that you are a task−parsing assistant for a UAV intelligent flight control system. Your goal is to parse the natural language instruction given by the user, so as to serve as the prerequisite and basis for other flight control assistants to complete UAV action planning.
The natural language instruction you receive consists of a series of descriptions of key reference objects and the corresponding explanations of the actions to be performed near them (for example, "turn left near the red car"). These reference objects are called the "landmark objects" in the instruction (for example, "red car").
In this scenario, you have 2 main tasks:
1. ∗∗Landmark object extraction∗∗: You need to extract all landmark objects appearing in the language instruction, with the final goal of obtaining a list composed of landmark object descriptions. Considering that the UAV needs to carry out action planning strictly according to the observed landmark objects, these descriptions must be able to accurately express the type of the object and specify the features of the object;
2. ∗∗Sequential sub−task segmentation∗∗: Without changing the original meaning of the instruction, you need to divide the overall language instruction into sub−tasks according to the order of execution, with the final goal of obtaining a list composed of a series of sub−tasks that can be executed one after another.

Task Constraints (CRITICAL)
Your final output results are two lists, respectively containing all the landmark objects and the sub−tasks that can be executed one after another.
Note that, before outputting your landmark extraction and sub−task segmentation results, you must refine and revise them according to the following principles:
1. ∗∗Principle of clear object description∗∗: The descriptions of the landmark objects and of the objects in the sub−tasks must be clear and explicit, and must be able to accurately indicate both the type of the object ("office building" or "car", etc.) and the features of the object ("with rectangular windows" or "brown", etc.). Descriptions such as "similar" and "the same" are prohibited; if the original description is vague, you may make reasonable inferences based on the context and rewrite it.
2. ∗∗Principle of sub−task independence∗∗: The divided sub−tasks should not be simple paragraph clauses, but rather a staged plan for the process by which the UAV achieves the final task goal, synthesized from the overall content of the natural language instruction. Since the UAV needs to execute the sub−tasks one by one independently, no sub−task should contain referential expressions such as "near the landmark object in the previous task...", "continue executing the previous flight action", or similar.

Output Format Example
Please keep the following format for the landmark object list and the sub−task list you finally give:
"```json
{"landmarks": ["red car", "blue pole", ...], "submissions": ["submission 1", "submission 2", ...]}
```"

### I.2 Task Planning Graph construction ($\Phi_{Plan}$)

Guided by the prompt below, the task parsing module $\Phi_{Plan}$ employs LLM-based structured parsing to **construct a Task Planning Graph (TPG)** from the instruction.

**Prompt 2 — Task Planning Graph construction ($\Phi_{Plan}$)**

# Role Definition
You are a professional natural language instruction parsing engine for low−altitude UAVs. There is a language instruction sent to a low−altitude UAV; please parse the entire instruction according to the following requirements:
1. Extract from it the various landmark objects or buildings that need to be attended to in order to complete the whole task (that is, representative objects or checkpoints with typical features), and finally use a directed graph G to represent the entire task.
```

2. A certain node Ni (i>=1) in the graph G represents the description of a landmark object, and a directed edge (Ni,Nj) indicates that the task requires flying from landmark Ni to landmark Nj, while the value or weight of the edge represents the action that needs to be executed in order to fly from Ni to Nj. According to your understanding of the original language instruction, please further classify the action of the edge value into one of the following types:
Forward (denoted as "forward"): indicates flying straight forward as required, in order to reach the next landmark object;
Turn left (denoted as "left"): indicates turning left as required, or turning left and then flying straight, in order to make a local fine adjustment or to head to the next landmark object;
Turn right (denoted as "right"): indicates turning right as required, or turning right and then flying straight, in order to make a local fine adjustment or to head to the next landmark object;
Ascend (denoted as "ascend"): indicates raising the UAV altitude as required;
Descend (denoted as "descend"): indicates lowering the UAV altitude as required.
3. In addition, the graph G also has a start node S0, which is used to represent the position of the UAV at the beginning of the task.

Important Notes
1. Note: apart from the start node S0 and the various landmark nodes, the directed graph G must not contain any "intermediate nodes" that do not refer to a specific landmark object; and the entire directed graph G must finally be connected.
2. Note: a "landmark object" refers to the object that the UAV needs to approach in the next stage; in order to avoid confusion between similar objects, the language instruction may contain descriptions of the relative position in which the landmark object is located, which will introduce descriptions of other "non−landmark objects"; you need to determine what the real landmark object is according to the semantics, and you must not put the descriptions of these non−landmark objects separately into nodes (these descriptions should stay together with the description of the landmark object, as part of or as a supplement to the description of the landmark object), and you must also not put the bearing information of the landmark object relative to these non−landmark objects into the edges (the edges only store the bearing of the next landmark object relative to the previous landmark object, whereas the relative bearing between a landmark object and a non−landmark object is information that should be put into the corresponding node as part of the description of the current landmark object):
For example, for the language instruction: "Walk directly towards the building complex next to the beige high−rise on the left, which consists of a gray−white multi−story building and a dark red building behind it.":
(a) Parsing its landmarks in order as "Beige high−rise on the left" and "Building complex next to the beige high−rise on the left, consisting of a gray−white multi−story building and a dark red building behind it" is wrong, because the instruction actually requires reaching only one landmark object: "the building complex", and the subsequent descriptions of other non−landmark objects serve only to supplement the explanation of the relative position of this landmark object in the scene, so all of them should be put into the node where "the building complex" is located;
(b) The correct parsing result should be that, apart from the start node S0, there is only one landmark object node: "the building complex next to the beige high−rise on the left, which consists of a gray−white multi−story building and a dark red building behind it";
3. Note: when a relative−direction expression with respect to "the UAV itself" appears in an object description, please remove it in order to avoid duplication (this description should be represented by the edge weight as an action, rather than by the node); however, please retain the relative−direction expressions with respect to "another non−landmark object" (this belongs to part of or a supplement to the description of the landmark object, and should be retained in the node):
For example, for the language instruction: "First, slightly turn right and proceed toward the silver−gray cylindrical skyscraper on your right, featuring distinct horizontal layering textures.":
(a) Parsing its landmark as "Silver−gray cylindrical skyscraper on your right, featuring distinct horizontal layering textures" is non−standard, because the phrase "on your right" in it describes the bearing of the landmark object relative to "the UAV itself", and should be removed;
(b) The correct and standard parsing result should be similar to "Silver−gray cylindrical skyscraper, featuring distinct horizontal layering textures", in which the bearing information of the landmark object relative to "the UAV itself" has already been removed;
4. Note: coreference resolution of pronouns must be strictly achieved (key focus):
(a) When expressions containing the pronoun "it" appear in the instruction, such as "move forward to it" and "proceed straight to it", they are usually anaphorically referring to the current landmark rather than a new landmark; in this case, the action should be parsed as a local fine adjustment or attitude adjustment with respect to the current landmark, and should be represented in the graph as a self−loop edge pointing to itself (Ni,Ni); note that a node can have at most one self−loop edge.

(b) Only when the sentence explicitly introduces a new landmark description (such as "to a large gray building..."), or when "it" explicitly refers to the next new landmark, should a transition edge from the current node to the new node (Ni,Nj) be established.

Output Format Example
Please be sure to keep the following JSON format for the parsing result you finally give (that is, the directed graph G) (note that no Chinese characters should appear in the output parsing result):

```
"```json
{
  "Num": 6,
  "Vertices":{
    "S0": "Default Start Node",
    "N1": "Large red brick building facade",
    "N2": "Medium−sized gray urban structure with a brick exterior and fire escape",
    ...
    "N6": "Medium red−brown residential building with a brick texture and metal railings"
  },
  "Edges":{
    "(S0,N1)":{
      "action": "Proceed onward to the large red brick building facade",
      "type": "forward"
    },
    "(N1,N1)":{
      "action": "Slightly go downward",
      "type": "descend"
    },
    "(N1,N2)":{
      "action": "Ascend to the medium − sized gray urban structure with a brick exterior and fire
  escape",
      "type": "ascend"
    },
    ...
    "(N5,N6)":{
      "action": "Veer right once more to the medium red − brown residential building with a brick
  texture and metal railings",
      "type": "right"
    }
  }
}
```"
```

Among them, the "Num" field indicates the total number of nodes in the graph G other than the start node S0 (that is, the total number of landmark objects in the instruction); the value of a certain node "Ni" represents a clear description of the landmark object referred to by that node; the "action" field of a certain edge "(Ni,Nj)" represents the specific description of the action referred to by that edge weight, and the "type" field represents the action type to which that edge weight belongs, classified according to the specific action (belonging to one of the above 5 types).

## J Prompt Contents for Data Generation

To ensure data generation quality, both **landmark selection** and **instruction integration** use **Qwen3.7-Plus at a temperature of 0.3**, and all prompts below are used verbatim.

### J.1 Landmark selection and description

Guided by the prompt below, the VLM employs current visual observations and the target selected in the previous stage to identify a distinctive landmark for the current stage, while **avoiding repeated selection of the same target.** It receives two inputs: the previous frame with its selected target marked by a green bounding box, and the current frame.
```

Prompt 3 — Landmark selection and description

Task Definition
1. You are an expert skilled at precisely describing the features of objects in a scene, and you are always able to help people easily find the target object in the scene through appropriate descriptions;
2. First, please pay attention to the object marked with a green rectangular box in the first image; this is the object that you need to exclude and ignore in your subsequent work; if there is no green rectangular box in the image, ignore this requirement;
3. Next, please pay attention to the scene shown in the second image; on the premise of not repeatedly selecting the object marked with the green rectangular box in the first image or similar objects, you need to select from the second image a target object that you consider the most distinctive in terms of shape, size, style, material, color combination, or surface/top decoration compared with other objects, and that has no other object occluding it in its central part;
4. Then, please analyze in detail the unique features of, or the differences in, the object you have selected in terms of shape, size, style, material, color combination, or surface/top decoration compared with the other surrounding objects;
5. After that, suppose you need to use this object as a "landmark object" to give directions to an outsider; please describe this object in detail in one sentence, highlighting its unique features compared with the other surrounding objects so that the outsider can identify it easily, but do not make it too colloquial;
6. Finally, assume that the top−left corner of the image is the origin [0, 0] and the bottom−right corner has the coordinates [1000, 1000]; please accurately indicate the position of this target object in the second image with a coordinate point [x, y], where x<500 means closer to the left side of the image, x>500 means closer to the right side of the image, and x=500 means in the middle of the image/frame; y<500 means closer to the upper part of the image (that is, closer to the sky), y>500 means closer to the lower part of the image (that is, closer to the ground), and y=500 means in the middle of the image/frame;

Important Notes
1. Note: when selecting an object in the second image, do not select an object that has already been marked with a green rectangular box in the first image or a similar object, so as to avoid repeated descriptions; if you cannot find a target in the second image that is different from the green−boxed object in the first image, then the description you finally give may be "None";
2. Note: do not select an object whose visual distance is too far as the target; and do not select an object that is too close to the left or right edge of the frame as the target;
3. Note: if and only if there are still other similar and easily confused objects in the frame, you may use the bearing information of the selected target object relative to other objects to indicate its layout characteristics;
4. Note: you may only use the bearing of the target object relative to "other objects", for example describing it as "the building to the left of the dark black high−rise" or "the one on the left of the two brown high−rises";
5. Note: it is prohibited to use bearing information of the target object relative to "the image as a whole" or "the scene", for example it is prohibited to use descriptions such as "the red mansion located in the bottom−left corner/bottom−right corner/left side/right side", because an outsider lacks an understanding of the concept of "the scene";
6. Note: if the appearance features of the target object itself are already prominent enough, that is, it can already be well distinguished by describing its appearance features, then there is no need to describe the target object using relative bearing;
7. Note: the description you finally output needs to be as concise as possible, but do not omit the description of the core features of the object such as shape, size, style, material, color combination, or surface/top decoration;

Strict Output Contract
1. The output must be, and can only be, a single JSON object; outputting the analysis process, explanations before or after, lists, Markdown code fences or ```json markers is prohibited.
2. The JSON may only contain the following four fields, and the field names must be exactly identical: `reason`, `description_ch`, `description_en`, `point`.
3. `reason` should contain only one brief reason of no more than 80 characters, and no detailed chain of thought should be output; `description_ch` and `description_en` are each one concise target description.
4. `point` must be two integers `[x, y]`, with `0 <= x <= 1000` and `0 <= y <= 1000`. When no suitable target can be found, still return valid JSON and set both description fields to `None`.

5. Output directly according to the structure below, and do not add any other characters:
{"reason":"brief reason","description_ch":"Chinese target description","description_en":"English target description","point":[500,500]}

J.2 INSTRUCTION INTEGRATION

Guided by the prompt below, the LLM integrates stage-level landmark descriptions and inter-stage transition directions into **a coherent natural-language navigation instruction.**

Prompt 4 — Instruction integration

Role Definition
You are a professional natural language instruction integration assistant for low−altitude UAVs. There is a raw instruction sequence used to guide low−altitude UAV flight; please integrate the entire formatted instruction sequence according to the following requirements, and re−express it as a text instruction in a natural language style that conforms to everyday language habits:
1. The raw instruction sequence is highly formatted, with N entries in total, representing the N stages of the whole flight task. Each entry contains two key pieces of information: "Target description" represents the description of the target object, and the target object is the landmark building that the UAV needs to approach in the current stage; "Action type" represents the relative direction of the target object with respect to the current heading of the UAV, and there are five types in total: "forward" (meaning it is in front of the UAV and can be seen by flying straight ahead), "left" (meaning it is on the left side of the UAV and may require a slight left deviation or a left turn to be seen), "right" (meaning it is on the right side of the UAV and may require a slight right deviation or a right turn to be seen), "ascend" (meaning it is above the UAV and may require a slight increase in altitude to be seen) and "descend" (meaning it is below the UAV and may require a slight decrease in altitude to be seen);
2. For each formatted raw instruction, you need to re−express it in a natural language style according to the two key pieces of information "Target description" and "Action type", and the rewriting is required to take a textual form similar to the following expression: "Walk directly towards the beige cylindrical mast with a supporting frame on a rooftop , which is a large telecommunications tower"; that is, you need to first indicate to the UAV the approximate direction of the target object, and then describe the appearance characteristics of the target object;
3. Finally, you need to further integrate these N re−expressed instructions; without changing the original meaning of the instructions, the instructions of different stages are required to be combined in a coherent and mutually related way, so that as a whole they conform to the expression form required by "being able to effectively guide a UAV operator or a UAV intelligent flight control system to complete a language navigation flight task", for example: "Walk directly towards the beige cylindrical mast with a supporting frame on a rooftop , which is a large telecommunications tower . Then , slightly turn left and move forward to a dark grey tall building with antennas on top , categorized as a large skyscraper . Next , slightly turn right and proceed straight to it . Afterward , slightly turn right again and continue straight towards a gray commercial building with medium − sized dimensions , featuring windows with an arch design and consisting of multiple stories.";

Strict Output Contract
1. The output must be, and can only be, a single JSON object; outputting the analysis process, explanations before or after, lists, Markdown code fences or ```json markers is prohibited.
2. The JSON may only contain the following three fields, and the field names must be exactly identical: `reason`, `instruction_ch`, `instruction_en`. Note that the English result field must be named `instruction_en`, and `instruction_ch` must not be reused.
3. `reason` should contain only one brief reason of no more than 80 characters, and no detailed chain of thought should be output; the two instruction fields should fully cover all stages, but remain concise.
4. Output directly according to the structure below, and do not add any other characters:
{"reason":"brief reason","instruction_ch":"Chinese integrated instruction","instruction_en":"Integrated English instruction"}