LookasideVLN: Direction-Aware Aerial Vision-and-Language Navigation

Supplementary Material

1. Spatial Landmark Knowledge Base

1.1. Landmark Recognizer

The Landmark Recognizer aims to identify visible landmarks present in historical observations. To this end, we leverage the strong image captioning capabilities of multimodal large language models to generate landmark descriptions from RGB observations. Specifically, we use Qwen-VL-Max [3] to extract landmark descriptions from each panoramic observation $O = \{o_i\}_{i=1}^{5}$ (without the upward view). The prompt used for the Landmark Recognizer is as follows:

Prompt for Landmark Recognizer

Task

Analyze the drone's multi-view images and list distinctive static landmarks using concise natural language descriptions. Follow these guidelines:

1. **Key Focus**:
   - Describe landmarks in several word phrases using this pattern (not strictly):
     `[Color] [Material] [Type] [Distinctive Feature]`
     (e.g., *"red brick water tower"*, *"blue ladder"*)
2. **View Processing**:
   For each view ('front/bottom/left/right/back'):
   - List navigation-critical objects, empty if none.
   - Use comparative references when helpful:
     (e.g., *"taller than surrounding buildings"*)

Example Output (JSON)

"front": [*"gray metal tower"*, *"blue bridge"*],
"bottom": [*"red-roofed circular building"*],
"left": [*"green glass skyscraper"*],
"right": [*"gray dome-shaped observatory"*],
"back": []

Input

The drone's multi-view images are:
- Front View: o_1,
- Bottom View: o_2,
- Left View: o_3,
- Right View: o_4,
- Back View: o_5

1.2. Extracting Landmark Descriptions

To obtain the landmark descriptions $\{l_i^{instr}\}_{i=1}^{N}$ used in Sec. 3.1.2 of the main paper, we develop a Landmark Parser that directly extracts landmark descriptions from the instruction $\mathcal{I}$ while preserving their original order. We implement the Landmark Parser using the large language model Qwen-Max [31]. The prompt used for the Landmark Parser is as follows:

Prompt for Landmark Parser

Task

You are a **Navigation Landmark Parser**. Your job is to:

1. **Identify every landmark reference** in the instruction, including:
   - **Street names**
     (e.g., *"Main Street"*, *"5th Avenue"*)
   - **Road types**
     (e.g., *"highway"*, *"dirt road"*)
   - **Intersections**
     (e.g., *"the corner of Park and Elm"*)
   - **Points of interest**
     (e.g., *"gas station"*, *"red mailbox"*)
2. **List them EXACTLY in order**
   including all prefixes/suffixes.

Output Format (JSON List)

["landmark_name_1", "landmark_name_2", ...]

Examples

- Instruction: *"Turn left on Maple Street, then right at the bank."*
  Output: [*"Maple Street"*, *"bank"*]
- Instruction: *"Follow Highway 1 until the second traffic light."*
  Output: [*"Highway 1"*, *"traffic light"*]

Input

The navigation instruction to process is " $\mathcal{I}$ ".

1.3. Landmark Localization with Bounding Box

In Eq. 3, we use the center pixel coordinates (u_i, v_i) of the bounding box b_i, along with the estimated depth $\bar{d}_i$, to compute the 3D position of the landmark l_i. We apply perspective projection to our depth observations to ensure consistency with the RGB inputs. In the following paragraphs, we detail the transformation from pixel coordinates

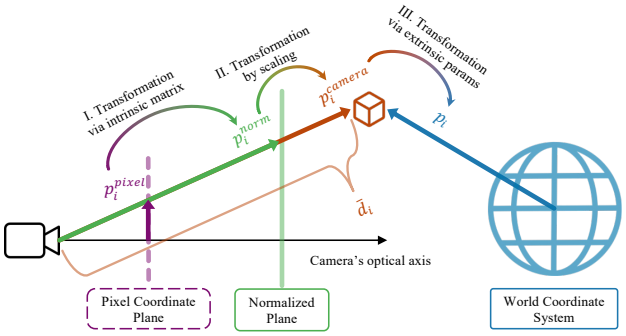

Figure 5. Illustration of the transformation from pixel coordinates to world coordinates under perspective projection

to world coordinates under perspective projection, as it involves more complex computations than those under orthographic projection.

As shown in Fig. 5, given the homogeneous pixel coordinate $p_i^{pixel} = [u_i, v_i, 1]^\top$, the corresponding position in normalized plane* is computed as:

$$p_i^{norm} = K^{-1} p_i^{pixel}, \tag{9}$$

where $K \in \mathbb{R}^{3\times3}$ denotes the camera's intrinsic matrix. The depth value $\bar{d}_i$ of the target landmark is defined as the Euclidean distance from the camera center to the object under perspective projection. Accordingly, the landmark's 3D position in camera coordinates can be represented as:

$$p_i^{camera} = \bar{d}_i \cdot \frac{p_i^{norm}}{||p_i^{norm}||_2}. \tag{10}$$

To transform this point into world coordinates, the camera's extrinsic parameters are applied:

$$\begin{aligned} p_i &= R p_i^{camera} + T \\ &= \frac{\bar{d}_i}{\|K^{-1} p_i^{pixel}\|_2} \cdot R K^{-1} p_i^{pixel} + T, \end{aligned} \tag{11}$$

where $R \in SO(3)$ and $T \in \mathbb{R}^3$ are the camera's rotation matrix and translation vector, respectively.

1.4. Pruning Strategy for SLKB

Since the agent may revisit the same location multiple times throughout its navigation history, the recorded observations can be overlapped. Thus, multiple detection results of the same landmark in nearby locations may appear in the Spatial Landmark Knowledge Base (SLKB)—*i.e.*, several locations in $\{p_{i,0}^{kb}, p_{i,1}^{kb}, \dots\}$ corresponding to the landmark description l_i^{kb} may refer to the same physical instance in the scene. To reduce redundancy, we apply a non-maximum suppression (NMS)-like pruning strategy based on the Euclidean distances between candidate locations. Specifically, for the candidate location list $\{p_{i,0}^{kb}, p_{i,1}^{kb}, \dots\}$ of each landmark description l_i^{kb} in the SLKB, we first sort the candidates by their associated confidence scores in descending order. Starting from the highest-scoring location, we select it as a representative detection and remove all other candidates within a predefined Euclidean distance threshold of 20 units, as these are considered duplicates. This process is repeated iteratively on the remaining candidates until no overlapping locations remain. The result is a refined set of landmark positions with minimal spatial redundancy, improving the accuracy and efficiency of the knowledge base.

*The normalized image plane is a virtual plane placed at a unit distance in front of the camera and perpendicular to its optical axis.

2. Pruning Strategy for ELG

Since large-scale urban scenes often contain many visually similar landmark instances that can be grounded by the same landmark description, traversing the entire graph can lead to redundancy. To address this, we adopt a pruning strategy to eliminate redundant nodes and edges in the ELG. Specifically, given the ordered set of unvisited landmarks and their candidate positions, $\mathcal{L}^{unvis} = \{l_i^{unvis} : \{p_{i,0}^{unvis}, p_{i,1}^{unvis}, \dots\}\}_{i=1}^{N_{ahead}}$, we retain the top $N_{next} = 6$ candidate positions closest to the agent's current location for the first unvisited landmark l_1^{unvis}. For each candidate position $p_{i,j}^{unvis}$ of the i-th unvisited landmark l_i^{unvis}, we connect it to the $N_{subseq} = 2$ nearest candidate positions of the subsequent landmark l_{i+1}^{unvis}, i.e., nearest positions selected from $\{p_{i+1,m}^{unvis}\}_{m=1}^{\dots}$.

3. Implementation of the Lookaside MLLM Navigation Agent

We build the agent with a chain-of-thought mechanism for robust and explainable path planning. Specifically, we design three types of prompts to handle different situations the agent may encounter during navigation: 1) *When the Egocentric Lookaside Graph (ELG) is available*, the agent selects a navigation path by reasoning over future subgoals and their spatial relationships. 2) *When a path has been selected and the next candidate landmark is known*, the agent determines the next action based on the landmark's location. This action is further refined using the current observation and the navigation instruction. 3) *In rare cases where the instruction contains no identifiable landmark*, the agent navigates by relying solely on visual observations and the navigation instruction. For each of these situations, we craft dedicated prompts for the agent to enable effective and robust reasoning. Moreover, the agent maintains a record of visited landmarks and summarizes its navigation history in text to support contextual reasoning. Below, we outline the prompt strategies for each scenario.

When the ELG is available, the agent is guided to reason

over candidate paths and predict the next action accordingly. The prompt used at this stage is as follows:

Prompt with ELG

System Message

You are a visual-and-language navigation agent that follows navigation instructions to move to each specific landmark and finally reach the destination.

Input Description

- **Instruction**: A natural language navigation instruction that guides the agent to reach some specific landmarks.
- **History**: A record of the landmarks that the agent has visited with a brief trajectory description.
- **Visited Landmark**: A list of landmarks that the agent has visited.
- **Candidate Paths**: A set of candidate navigation paths.
- **Current Views**: Six views observed at the current time step: front, bird's-eye, left, right, back, and upward views.

Output Description

- **Observation Description**: The current observation and the landmark you seem based on the current views.
- **Navigation Progress**: Describe the current navigation progress based on the instruction and the history.
- **Reasoning for Path Selection**: Identify unvisited landmarks in the Candidate Paths to visit next. Give a description on how to reach the unvisited landmarks based on the instruction step by step with the relevant instruction snippets. For example, "Next Two Landmarks: A and B. Instruction Snippet: *Turn left at A, then move forward to B*."
- **Selected Path ID**: Based on the reasoning above, select one of the candidate paths to follow by providing its ID. If you choose to create your own plan, respond with -1.
- **Action Reasoning**: Given the selected path, determine the next action toward the next landmark. Refine your decision using the navigation instruction and current visual observation.
- **Action**: Choose the next action from {*forward*, *turn_left*, *turn_right*, *ascend*, *descend*, *stop*} The *forward* action will let you move **5 meters** in current direction; A turning action will result in a **15 degree** turning; The *ascend* / *descend* action will cause an altitude change of **2 meters**; The stop action should be triggered when the final landmark is reached.
- **Calculation for Number of Executions**: For example, executing a turning action six times at 15 degrees per turn will result in a total rotation of 15 x 6 = 90 degrees.
- **Number of Executions**: The number of times this action should be executed.
- **Updated History**: Update the history according to the landmark you see and the action you take.

After selecting a path, the agent determines the next action based on the spatial relationship to the upcoming landmark, further refined using current observations and instructions. The prompt at this stage is adapted from the previous version by replacing "Candidate Paths" with "The Next Landmark" in the input. Correspondingly, the path selection reasoning is replaced with landmark-based reasoning. The prompt used at this stage is as follows:

Prompt with the next landmark

System Message ...

Input Description

...

- **The Next Landmark**: The upcoming landmark the agent needs to reach, along with its relative location.

...

Output Description

...

- **Reasoning for Landmark to Follow**: Reason whether to proceed toward the next landmark by considering the given instruction, current observation, and the landmark's relative position.
- **Whether to Follow the Next Landmark**: Select whether to follow the next landmark or not. Respond with *follow* or *not follow*.

...

In rare cases where instructions lack identifiable landmarks, the agent relies on general spatial understanding to determine navigation direction. Unlike previous prompts, information about candidate paths and landmark locations is excluded from the input.

4. More Experiment Results

4.1. Additional MLLM ablations.

We add GPT-series and Qwen3VL as MLLM backbones in Tab. 6. We note that Qwen2.5VL-32B is specifically

Table 6. Additional ablation study on different MLLMs.

Category	Method	SR↑	SDTW↑	NE↓
Qwens	Qwen-2.5-VL-7B	9.0	1.7	306.6
	Qwen-3-VL-8B	11.7	3.8	114.1
	Qwen-2.5-VL-32B	14.1	4.9	94.3
	Qwen-3-VL-32B	11.7	5.1	84.8
GPTs	GPT-4V	15.0	6.6	80.8
	GPT-4o	15.3	6.3	83.5

optimized with RL[*], which explains its stronger performance compared to Qwen3VL-32B, while GPTs achieve the best overall results, further validating the effectiveness of LookasideVLN.

4.2. Sensitivity analysis on depth noise.

We conduct a sensitivity analysis in Tab. 7 by adding Gaussian noise to the depth $\bar{d}_i$ in Eq. 3. LookasideVLN remains robust under large noise (up to $\sigma = 5$ meters), as the agent can still infer landmark directions from its observations.

Table 7. Sensitivity analysis on the seen split of AerialVLN-S.

Depth Noise Std. (σ)	SR↑	SDTW↑	NE↓
$\sigma = 0$ meters	14.7	5.4	77.1
$\sigma = 5$ meters	13.2	4.7	83.5
$\sigma = 20$ meters	11.1	3.5	84.4

4.3. Qualitative Examples of Reasoning Process

Fig. 6 shows more examples of the chain-of-thought reasoning. The agent successfully grounds instructions with path descriptions derived from the ELG and leverages the egocentric lookaside directional relationships for robust path selections.

Fig. 7 shows key steps of a navigation episode where LookasideVLN follows complex instructions through an urban environment. Starting with textual instructions, the agent successfully navigates through multiple waypoints, maintaining correct visual perceptions. The sequence demonstrates the agent's ability to interpret visual scenes and make appropriate navigation decisions to reach the target destination.

4.4. Failure Cases

Fig. 8 presents a failure case in the reasoning process. The instruction includes two consecutive directional cues—"*turn slightly left*" and "*turn around*". The agent is misled by the former and fails to attend to the latter, resulting in an incorrect path selection.

[*]Refer to https://qwen.ai/blog?id=qwen2.5-vl-32b

5. Limitations and Future Work

Although LookasideVLN achieves state-of-the-art performance in the simulated environment, it has not yet been deployed in the real world. Future work may focus on bridging the sim-to-real gap and evaluating the system on real UAVs. Additionally, improving robustness under real-world conditions will be an important direction. A detailed discussion of the limitations and future work is provided below.

5.1. Limitation Discussion

Although LookasideVLN outperforms most prior state-of-the-art baselines on the challenging AerialVLN benchmark, the overall performance remains limited. The reasons are twofold:

1. Learning-based baselines are constrained by the scarcity of annotated trajectories and the limited scale of training scenes. While they can perform accurate action selection (at each step) during end-to-end training, they tend to accumulate errors (over the entire navigation process) in evaluation and struggle to generalize to unseen environments. This issue may be alleviated through more cost-efficient data collection strategies and dedicated error-correction mechanisms.
2. Zero-shot LLM-based approaches are strong in understanding natural language instructions and visual observations. However, they struggle to comprehend structured 3D scenes when relying solely on language and RGB inputs, lacking access to fine-grained 3D information such as depth cues. They also face challenges in ego-motion understanding—often misinterpreting ego-trajectories when processing egocentric navigation videos. These limitations may be mitigated by leveraging more advanced MLLMs specialized in 3D scene comprehension and ego-motion modeling.

5.2. Future Work

We validate LookasideVLN in simulated environments but have not yet conducted real-world evaluations. In the AerialVLN benchmark, the agent is instructed to output a single action at each step to align with the benchmark's evaluation settings. However, invoking the MLLM-based agent for reasoning at every step is computationally expensive and cannot be directly deployed on real-world UAVs. In practical deployment, prompting the agent to generate planned waypoints along the selected landmark-level path within the ELG—and executing these waypoints using low-level motion planners—could reduce reasoning time while enabling safety checks and occlusion avoidance.

Instruction: Take off and turn left to the stairs and turn left at them. Go over wires…
Path Descriptions:
[Plan id:0] Move forward 17.4 meters to reach stairs. Then turn left 15.1 degrees and move towards wires.
[Plan id:1] Move forward 17.4 meters to reach stairs. Then turn back and move towards wires.
[Plan id:2] Turn left 87.7 degrees, move forward 17.8 meters and ascend 5.2 meters to reach stairs. Then turn left 133.2 degrees and move towards wires.
[Plan id:3] Turn left 87.7 degrees, move forward 17.8 meters and ascend 5.2 meters to reach stairs. Then turn right 119.4 degrees and move towards wires.
[Plan id:4] Turn right 109.1 degrees, move forward 20.7 meters and ascend 6.5 meters to reach stairs. Then turn right 85.0 degrees and move towards wires.
Chain-of-Thought:
1. *Next Two Landmarks*: stairs and wires
2. *Instruction Snippet*: …
3. *Path Selection*: Plan id:2 involves turning left, moving forward, and ascending to reach the stairs, which aligns with the instruction.

Instruction: Take off rising the height turn around and proceed forward towards near the bicycle. Rising the height and move forward towards above the railway track and platform…
Path Descriptions:
[Plan id:0] Turn left 173.0 degrees, move forward 50.2 meters to reach bicycle. Then turn left 45.1 degrees and move towards railway platform.
[Plan id:1] Turn left 173.0 degrees, move forward 50.2 meters to reach bicycle. Then move forward towards railway platform.
[Plan id:2] Turn right 166.9 degrees, move forward 67.7 meters to reach bicycle. Then turn right 87.5 degrees and move towards railway platform.
[Plan id:3] Turn right 144.9 degrees, move forward 71.7 meters to reach bicycle. Then turn right 135.0 degrees and move towards railway platform.
Chain-of-Thought:
1. *Next Two Landmarks*: bicycle and railway platform
2. *Instruction Snippet*: …
3. *Path Selection*: Plan id:1 involves turning around (left 173.0 degrees) and moving forward to reach the bicycle, then continuing forward towards the railway platform, which aligns with the instruction

Instruction: Take off and turn left and move forward and over the subway and turn left and towards the bike…
Path Descriptions:
[Plan id:0] Turn right 80.9 degrees, move forward 6.9 meters and ascend 6.8 meters to reach subway. Then turn back and move towards bike.
[Plan id:1] Turn right 80.9 degrees, move forward 6.9 meters and ascend 6.8 meters to reach subway. Then turn left 107.8 degrees and move towards bike.
[Plan id:2] Turn left 100.0 degrees, move forward 28.6 meters and ascend 5.4 meters to reach subway. Then turn left 24.4 degrees and move towards bike.
[Plan id:3] Turn right 169.0 degrees, move forward 38.2 meters and ascend 3.0 meters to reach subway. Then turn right 60.5 degrees and move towards bike.
Chain-of-Thought:
1. *Next Two Landmarks*: subway and bike
2. *Instruction Snippet*: …
3. *Path Selection*: Plan id:2 seems to align well with this requirement as it involves turning left and moving forward to reach the subway.

Figure 6. Examples of the agent's reasoning. The agent selects optimal paths using descriptions derived from the Egocentric Lookaside Graph. Highlighted words indicate successful path decisions that align with directional cues in the instructions.

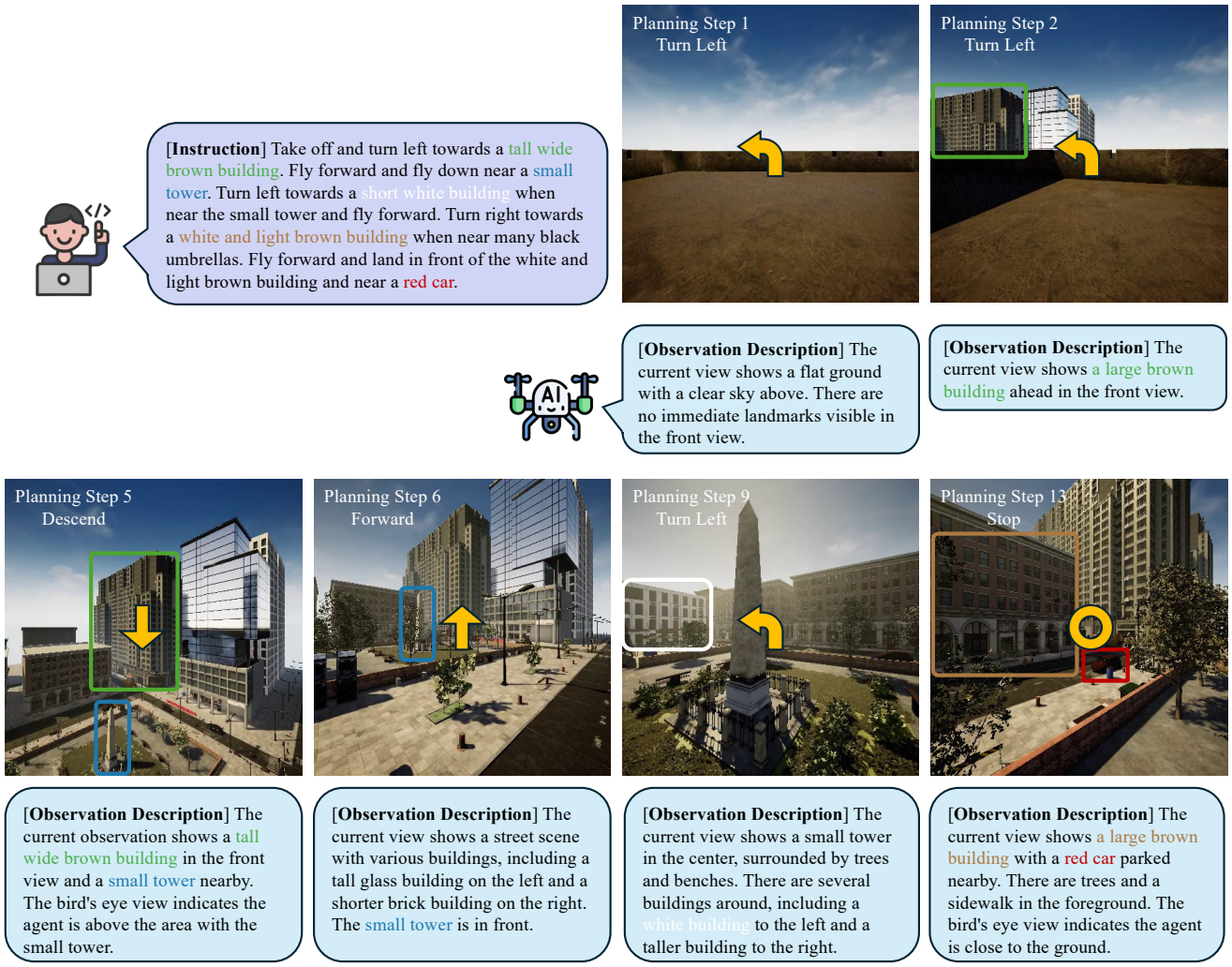

Figure 7. Visualization of key steps in a navigation episode. LookasideVLN provides accurate observation descriptions and makes appropriate action decisions.

Instruction: Take off and fly straight toward the blue roof with wood boards laying on top of it. Turn slightly left and turn around facing a large tree…
Path Descriptions:
[Plan id:0] Turn left 58.2 degrees, move forward 56.6 meters and ascend 13.0 meters to reach roof with blue. Then turn back and move towards large tree.
[Plan id:1] Turn left 58.2 degrees, move forward 56.6 meters and ascend 13.0 meters to reach roof with blue. Then turn left 34.4 degrees and move towards large tree.
Chain-of-Thought:
1. *Next Two Landmarks*: blue roof and large tree
2. *Instruction Snippet*: …
3. *Path Selection*: Plan 1 aligns better with the instruction as it involves turning left after reaching the blue roof, which matches the instruction to 'turn slightly left'.

Figure 8. An example of failure cases where the agent's reasoning process fails due to hallucinations. The agent correctly identifies the directional cue "slightly left" but fails to recognize the subsequent "turn around" instruction, leading to failed path planning.