

# FineCog-Nav: Integrating Fine-grained Cognitive Modules for Zero-shot Multimodal UAV Navigation

## Supplementary Material

### Contents

<b>1. Dataset Details</b>	<b>1</b>
1.1. Analysis of Issues in AerialVLN Dataset . . .	1
1.2. Detailed Construction Process . . . . .	1
1.3. Dataset Statistics . . . . .	3
<b>2. More Analysis</b>	<b>4</b>
2.1. Demo Video . . . . .	4
2.2. Qualitative Comparison with Baselines . . .	4
2.3. Quantitative Analysis of Ablations . . . . .	5
2.4. Qualitative Analysis of Ablations . . . . .	6
<b>3. Human Study Details</b>	<b>8</b>
3.1. Design of the Human Study Process . . . . .	8
3.2. Analysis of Human Study Results . . . . .	9
<b>4. More Evaluations</b>	<b>10</b>
4.1. Controlled Evaluation . . . . .	10
4.2. Generalization Evaluation . . . . .	10
4.3. Efficiency Evaluation . . . . .	11
4.4. Ablation of Collision Warning . . . . .	11
<b>5. Detailed Prompts for BaseModel</b>	<b>12</b>
5.1. Perception . . . . .	12
5.2. Action . . . . .	13
<b>6. Detailed Prompts for Framework Baselines</b>	<b>15</b>
6.1. NavGPT . . . . .	16
6.2. DiscussNav . . . . .	18
<b>7. Detailed Prompts for FineCog-Nav</b>	<b>19</b>
7.1. Instruction Parser . . . . .	19
7.2. Subgoal Extractor . . . . .	20
7.3. Attention . . . . .	21
7.4. Perception . . . . .	21
7.5. Imagination . . . . .	21
7.6. Subgoal Judger . . . . .	22
7.7. Multi-Level Memory . . . . .	23
7.8. Decision Making . . . . .	24
<b>8. Execution Details</b>	<b>24</b>
8.1. Step 0 . . . . .	25
8.2. Step 13 . . . . .	27
8.3. Step 14 . . . . .	30

### 1. Dataset Details

#### 1.1. Analysis of Issues in AerialVLN Dataset

While AerialVLN serves as a challenging and milestone benchmark for UAV VLN, we have observed a substantial proportion of flawed data samples during experimental analysis. To systematically analyze these issues, we randomly sampled 200 trajectory-instruction pairs from the AerialVLN-S subset and conducted a careful manual inspection. The issues are categorized into four main types, as illustrated in Fig. 1:

➡ **Trajectory-Instruction Misalignment** is the most prevalent issue, affecting nearly half of the samples. Such as the instruction specifying a left turn while the corresponding ground-truth trajectory turns right;

➡ **Collision Anomalies** rank second, comprising physically implausible trajectories that involve minor clipping (e.g., through fences/lampposts) or major clipping into walls/ground;

➡ **Ambiguous Instructions** manifest as two subtypes: a) Instructions lacking spatial or landmark references, leaving agents uncertain about environmental interactions post-action; b) Vague landmark descriptions (e.g., generic terms like “building” or “tree”) that lack distinctive visual correspondence in outdoor aerial scenes.

➡ **Invisible Landmarks** mean that landmarks mentioned in instructions as references are visually absent or unobservable in corresponding viewpoints, severely hindering navigation grounding.

#### 1.2. Detailed Construction Process

To ensure the high quality and representativeness of AerialVLN-Fine, we designed a systematic filtering and re-annotation pipeline that combines automated checks with manual verification, as shown in Fig. 2. This process improves data purity, completeness, and annotation consistency. The main steps are as follows:

① **Manual Review and Selection:** Each instruction-trajectory pair is manually examined against four key quality criteria. Erroneous or incomplete samples are removed, and for each scene, an average of 20 diverse and high-quality trajectories are retained to ensure both diversity and balance.

② **Instruction Segmentation and Refinement:** Complex instructions are semantically decomposed and numbered. Each sentence is checked for clear actions and termination conditions. Ambiguous or missing actions and landmarks

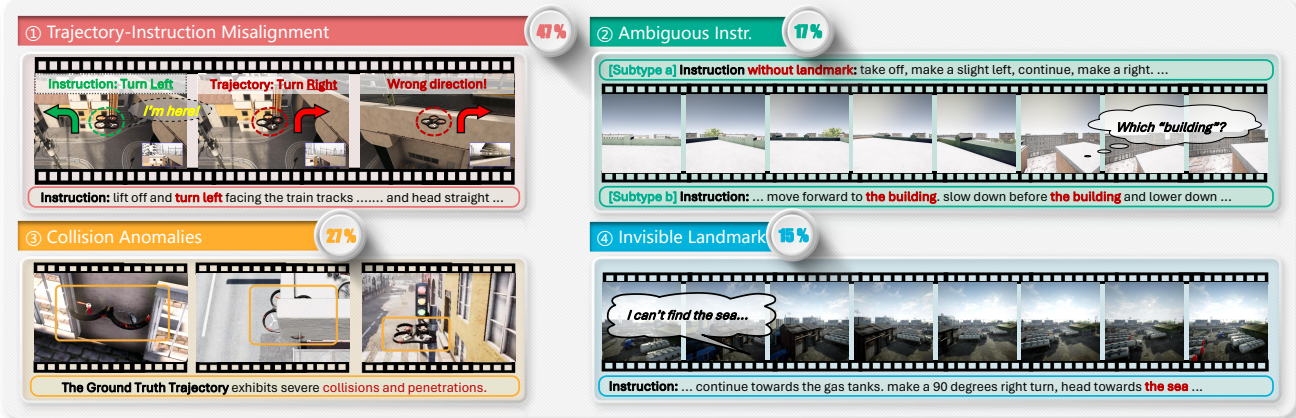


Figure 1. Through manual analysis of 200 randomly sampled instruction-trajectory pairs, we quantify four prevalent issues: → Trajectory-Instruction Misalignment (47%); → Collision Anomalies (27%); → Ambiguous Instructions (17%), and → Invisible Landmarks (15%).

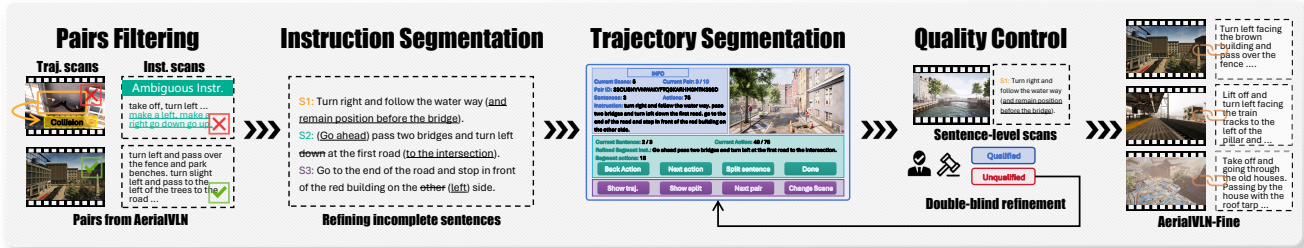


Figure 2. The construction process of AerialVLN-Fine, including pairs filtering, instruction segmentation, trajectory segmentation, and double-blind quality control to ensure high-quality, sentence-level annotations.

Table 1. Detailed statistics of AerialVLN-Fine by scene.

Scene ID	# Traj.	Path Len	# Action	# Sent.
2	16	279.19	92.13	5.19
3	32	121.91	72.47	4.56
5	10	315.80	128.20	6.20
7	30	210.83	71.70	4.70
8	35	103.26	47.40	4.29
9	30	255.50	80.03	4.20
10	10	334.60	115.50	5.20
11	20	413.10	125.45	4.65
12	20	167.65	67.05	4.95
13	32	98.69	62.16	4.25
14	17	137.59	70.06	4.53
17	30	96.17	58.87	4.07
20	2	425.50	143.00	7.50
21	4	265.50	80.00	3.50
24	12	138.67	82.67	5.58

are clarified or added based on the context and trajectory. Visual anchors along the trajectory are explicitly described

to guide the navigation. Key principles include:

- Each action is paired with a clear stopping condition, typically linked to a visible landmark (e.g., “proceed to the truck”, “turn left to face the graffiti”).
- Landmark descriptions are made unique and unambiguous, using type, color, text, or surrounding context as needed.
- Landmarks are ensured to be visible; if not, a visible anchor is added.
- Instructions are written in the first person and kept concise, covering actions such as move forward, turn left/right, ascend, or descend.

**③ Trajectory Segmentation and Alignment:** Trajectories are finely segmented according to the termination conditions of each sub-instruction, aligning each with its corresponding waypoint and action sequence. This results in more granular instruction-to-trajectory mappings.

**④ Rigorous Quality Control:** After annotation, multiple rounds of manual review are performed based on the quality criteria to further eliminate errors.

Through this process, AerialVLN-Fine achieves high accuracy, consistency, randomness, and trajectory realism,

Table 2. Qualitative Analysis of Sentence-Level Instructions

Capability Dimension	Description	Example	Count	Proportion/%
Spatial Relations	Understanding and processing relative positional relationships	Land to the right of the van that looks like the mystery machine from Scooby-Doo.	1387	99.36
Multi-Target Path Planning	Handling path planning between multiple locations	At the end of the road go straight across <b>the canal</b> then turn left facing <b>the road with bridges</b> on the left and head straight.	736	52.72
Trajectory Constraints	Following specific trajectory shapes or restrictions	Lower flight height and fly in a slow complete circle and maintain position. Maintain 5m altitude near the red warehouse.	564	40.40
Temporal Relations	Handling time-based task sequences	<b>Turn left after lifting off</b> and face the dog mural then head straight.	413	29.58
Ordinal/Cardinal Recognition	Understanding and processing ordinal or cardinal information in instructions	Turn around at <b>the fourth white building</b> in your path.	207	14.83

covering a broad range of scenarios. This provides a solid foundation for fair algorithm comparison and capability assessment, supporting in-depth research and innovation in UAV Vision-and-Language Navigation.

### 1.3. Dataset Statistics

In Tab. 1, we display detailed statistics of the AerialVLN-Fine dataset for each scene, including the number of trajectories, average path length, average number of actions, and average number of sentences per trajectory. We can see that the AerialVLN-Fine dataset exhibits several notable characteristics:

① First, it encompasses a diverse set of 15 scenes, providing a wide range of environments for evaluation;

② Second, the navigation trajectories are considerably **longer** and more **complex** compared to those in existing indoor VLN benchmarks. For example, while the R2R dataset features an average path length of approximately 10 meters with around 5 actions per trajectory, and RxR reports an average path length of 15 meters and 8 actions, the latest OctoNav-Bench includes trajectories averaging 8.2 meters in length (with the longest reaching 42 meters). In stark contrast, trajectories in AerialVLN-Fine require a substantially greater number of actions and cover much longer distances on average—specifically, 187 meters in path length and 76 actions per trajectory, underscoring its significantly higher navigation complexity.

③ Additionally, there is substantial variation in both path length and number of actions across different scenes, with some environments presenting significantly more challenging navigation tasks. Furthermore, the average number of sentences per trajectory generally ranges from 4 to 7, which

can be interpreted as the number of sub-tasks or instructions required to complete each navigation episode.

In Fig. 3, we show the distribution and demonstration of scenes in AerialVLN-Fine. The dataset exhibits rich environmental diversity and realistic perceptual challenges, which can be summarized as follows:

✿ **Urban areas:** Scenes such as *Abandoned City* (Scene 5), *Nighttime City* (Scene 11), *Rail Station* (Scene 17), and *Hong Kong City* (Scene 24) feature dense high-rise buildings, complex street layouts, and varying levels of urban decay. These settings test navigation robustness in cluttered metropolitan areas.

✿ **Rural and suburban areas:** *Abandoned Rural Area* (Scene 2), *Southern Village* (Scene 8), *Industrial Zone* (Scene 9) and *Warehouse Area* (Scene 12) represent low-density settlements with sparse landmarks, irregular structures, and natural terrain, posing challenges in localization due to limited visual cues.

✿ **Low-light and adverse weather conditions:** *Nighttime City* (Scene 11) and *Container Wharf* (Scene 21) are nighttime environments, while several scenes—including *Abandoned City* (Scene 5) and *Abandoned Port* (Scene 20)—contain fog or overcast conditions. These scenarios severely degrade color fidelity and landmark visibility.

✿ **Visual occlusion and ambiguous landmarks:** Dense foliage in *Business District* (Scene 3) and *Town Square* (Scene 14), combined with collapsed structures in *Abandoned Rural Area* (Scene 2) and *Abandoned City* (Scene 5), leads to severe view obstruction. Moreover, repetitive architectures in urban and industrial zones cause landmark ambiguity and confusion.

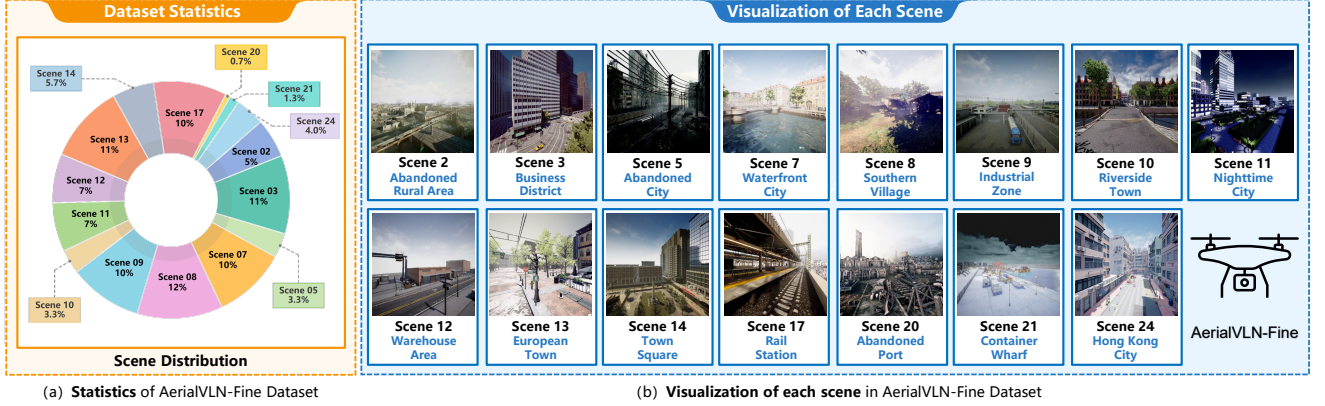


Figure 3. Distribution and Demonstration of scenes in AerialVLN-Fine. 15 scenes cover day, night, city, rural areas, and contain perceptual challenges such as similar landmarks and visual obstructions.

In addition, Tab. 2 presents a qualitative analysis of the capability dimensions reflected in sentence-level instructions within the AerialVLN-Fine dataset. The analysis covers five key capability dimensions: spatial relations, multi-target path planning, trajectory constraints, temporal relations, and ordinal/cardinal recognition. The results show that spatial relations account for the vast majority of instructions (99.36%), indicating that most navigation tasks require the agent to understand and reason about relative positional relationships. Multi-target path planning is the second most common capability, appearing in 52.72% of the instructions, reflecting the frequent need to navigate through multiple intermediate locations. Trajectory constraints are also prevalent, occurring in 40.40% of the instructions, which suggests common requirements for specific flight patterns or altitude restrictions. Temporal relations are present in 29.58% of cases, highlighting the importance of time-ordered actions, while ordinal/cardinal recognition appears in 14.83% of instructions, indicating a moderate need for understanding numerical cues. These findings highlight the diversity and complexity of navigation instructions in AerialVLN-Fine, reflecting a wide range of real-world challenges that test an agent’s ability to comprehend and execute various types of navigation tasks. We present a capability-aware ablation study in Sec. 2.3, where we systematically evaluate how each cognitive module contributes to these distinct reasoning demands.

## 2. More Analysis

### 2.1. Demo Video

To provide a more comprehensive and intuitive understanding of the navigation process, we additionally include a demonstration video in the supplementary material, to which we kindly refer interested readers for further insight. In this demo, we showcase not only the full navigation tra-

jectory of our proposed **FineCog-Nav**, but also compare it against the single-module baseline (*BaseModel*) and two framework-level baselines (*NavGPT* and *DiscussNav*) on representative navigation tasks with natural language instructions, sampled from the AerialVLN-Fine dataset.

As illustrated in the demo, FineCog-Nav successfully accomplishes the navigation task, faithfully following the provided instructions throughout the trajectory. In contrast, the baseline methods demonstrate a variety of failure modes, such as early deviation from the intended route, neglecting instructions, traversing through obstacles, and aimlessly navigating towards irrelevant areas (e.g., the sky or sea), among other issues.

### 2.2. Qualitative Comparison with Baselines

Since navigating UAVs in 3D continuous environments for Vision-and-Language Navigation (VLN) tasks remains highly challenging, existing zero-shot methods based on large language and vision-language models (LLMs and VLMs) generally achieve low success rates. Therefore, we also provide a qualitative analysis of failure cases to demonstrate that, even under these difficult conditions where all methods fail to reach the success rate threshold, our FineCog-Nav model exhibits markedly superior behavior.

As illustrated in Fig. 4, we examine a challenging episode with the following instruction (ID: 3WJE-QK0XAH0G632K1XLQY3JARLMA18):

*“Move forward along the road until reaching the bikes on left, slightly move left to the bikes. Move forward to the T-junction, then turn right to face the tunnel, and go through it. At the end of the tunnel land on the left side on the sidewalk.”*

We can see that FineCog-Nav demonstrates a strong ability to follow complex instructions and to perceive and utilize environmental cues. Specifically, our model correctly

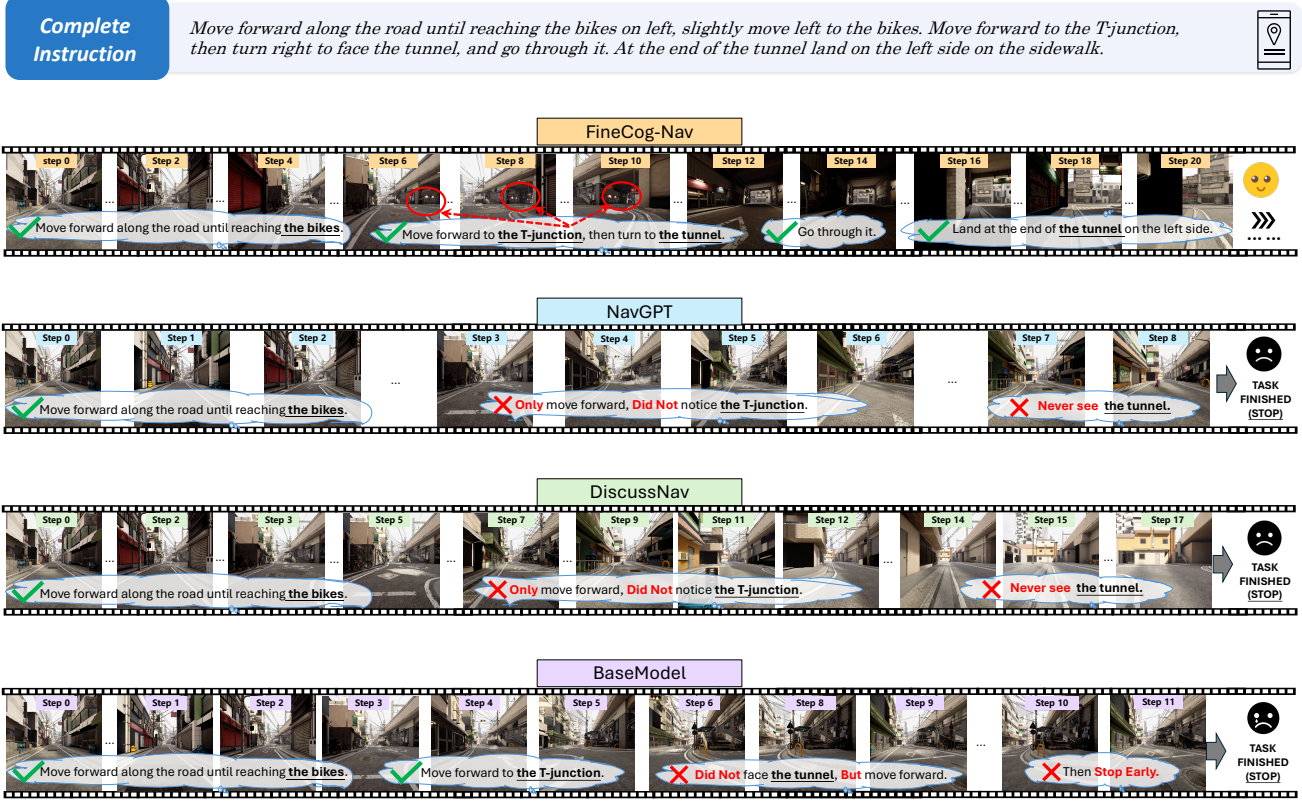


Figure 4. Qualitative comparison of FineCog-Nav and baselines on a challenging UAV VLN episode. FineCog-Nav demonstrates superior instruction-following by correctly identifying the key landmarks (the T-junction, the tunnel), while baseline methods overlook key landmarks, terminate prematurely, or encounter navigation errors.

detects the bikes as an initial landmark, then successfully navigates toward and recognizes the T-junction, and further identifies and enters the tunnel as instructed. Notably, FineCog-Nav is the only method to correctly complete the critical step of reaching the tunnel entrance, traversing it, and finally landing at the instructed location on the sidewalk. This sequence showcases the model’s precise step-by-step alignment with the provided instructions and robust landmark grounding throughout the navigation episode.

In contrast, the baseline methods show pronounced limitations:

- ◆ **NavGPT** fails to notice the T-junction, continuing straight past the intended turning point. Lacking awareness of the tunnel, NavGPT never attempts to enter it and terminates the task after only 8 steps.

- ◆ **DiscussNav** also overlooks the T-junction and continues without attempting to locate the tunnel, resulting in premature task termination before reaching the intended goal.

- ♠ For the single-module baseline (**BaseModel**), the model similarly fails to ground the instruction to “face the tunnel,” proceeds straight ahead without adjustment, and ultimately stops early.

In summary, these qualitative observations highlight that, although all methods face substantial challenges in the UAV VLN task, our FineCog-Nav stands out in its ability to consistently ground on-step instructions, accurately perceive and act upon key environmental landmarks, and maintain coherent progress towards the target. In contrast, baseline models frequently miss crucial cues, lose alignment with the instructions, and terminate navigation without fulfilling the task.

### 2.3. Quantitative Analysis of Ablations

We present further quantitative ablation analysis to elucidate the functional contributions of each module in FineCog-Nav. To understand how each component contributes to distinct capabilities, we analyze the 3D success rate ( $SR^{3D}$ ) across the five capability dimensions defined in Sec. 1.3: *spatial relations*, *multi-target planning*, *trajectory constraints*, *temporal relations*, and *ordinal/cardinal understanding*.

As summarized in Tab. 3, removing any single module degrades performance, with the degree and affected dimensions reflecting the specialized function of each component:

Table 3. **Per-Capability Ablation Analysis on  $SR^{3D}$  (%)**. The last row shows absolute Success Rates of the full model; other rows show  $\Delta SR^{3D}$  relative to full. For each ablation, the largest and second-largest performance drops are marked with bold and underline, respectively.

Module			Capability Dimension (SR % or $\Delta SR$ %)				
Attention	Imagination	Subgoal	Spatial Relations	Multi-Target Path Planning	Trajectory Constraints	Temporal Relations	Ordinal/Cardinal
✗	✓	✓	<u>-3.00</u>	-2.85	-2.75	-2.60	<b>-4.08</b>
✓	✗	✓	-2.33	<u>-2.49</u>	<b>-2.75</b>	-0.52	-2.04
✓	✓	✗	<u>-4.00</u>	-3.91	<b>-4.71</b>	-2.08	-3.40
✗	✗	✓	-2.33	-2.49	<b>-3.53</b>	<u>-3.13</u>	-2.72
✓	✗	✗	-2.00	<u>-2.14</u>	<b>-2.75</b>	-1.04	-2.04
✓	✓	✓	6.00	5.69	6.27	4.69	4.76

✗ **Hierarchical Memory:** Critical Infrastructure. Consistent with our main results, ablating the *Hierarchical Memory* module (replacing it with a simple flat history buffer) causes the largest and most pervasive performance drops across all capability dimensions (e.g.,  $-5.33$  for spatial relations,  $-5.49$  for trajectory constraints, and  $-4.69$  for temporal relations). This underscores that memory serves not as an auxiliary module, but as critical infrastructure that underpins and interconnects all reasoning components. As its impact is system-wide and not independent, we omit memory ablation results from the table. To elaborate a little, the hierarchical structure is especially effective for maintaining long-term contextual coherence, enabling robust information integration and planning over extended trajectories. Its absence disrupts both individual module effectiveness and overall system synergy, further validating our design.

✗ **Attention:** Ablating the *Attention* leads to the sharpest decline in *ordinal/cardinal understanding* ( $-4.08$ ), indicating that this module is essential for grounding numerical and sequential cues in language; without robust attention, the agent struggles with fine-grained language-visual alignment and stepwise reasoning.

✗ **Imagination:** Ablating *Imagination* results in a notable drop in *trajectory constraints* ( $-2.75$ ), and also affects *spatial relations* and *multi-target planning* to a lesser extent. This indicates that Imagination is crucial for simulating potential outcomes and evaluating subgoal feasibility, which is indispensable for satisfying path constraints and planning multi-stage routes.

✗ **Subgoal:** Ablating the *Subgoal* mechanism causes the most significant drops in *spatial relations* ( $-4.00$ ), *multi-target planning* ( $-3.91$ ), and *trajectory constraints* ( $-4.71$ ), reflecting its central role in decomposing complex instructions and structuring long-horizon navigation. The inability to break a task into manageable subgoals leads to compounding errors and loss of strategic planning.

Compound ablations further highlight the interdependence among modules:

✗ **Attention & Imagination:** Ablating both modules

leads to the greatest drops in *trajectory constraints* ( $-3.53$ ) and *temporal relations* ( $-3.13$ ), indicating that combining perceptual grounding with prospective scene imagination is critical for satisfying complex spatial constraints and ensuring the correct temporal ordering and execution of navigation subgoals.

✗ **Imagination & Subgoal:** Ablating these two modules causes severe degradation in *trajectory constraints* ( $-2.75$ ) and *multi-target planning* ( $-2.14$ ), as *Subgoal* provides structured decomposition and *Imagination* evaluates transition feasibility. Removing both disrupts the planning-verification loop that is central to coherent long-horizon navigation.

In summary, the above quantitative analysis demonstrates that the modules in FineCog-Nav fulfill complementary roles within the cognitive workflow, and their integration is crucial for achieving robust performance across the diverse and challenging scenarios in UAV VLN tasks.

## 2.4. Qualitative Analysis of Ablations

We qualitatively examine how ablating core modules affects agent behavior by visualizing representative examples, focusing specifically on the Hierarchical Memory and Attention modules.

✎ **Analysis of Hierarchical Memory.** Fig. 5 illustrates both the performance consequences of replacing hierarchical memory with a plain history buffer, and the differences in memory representations at a critical navigation stage.

① *Performance Impact:* Ablating hierarchical memory impairs the agent’s ability to execute complex instructions. With only a flat history buffer, the agent struggles to detect subgoal completion, delays subgoal switching, and makes navigation errors. In contrast, hierarchical memory maintains coherent progress tracking and enables timely, accurate subgoal transitions.

② *Memory Content Comparison:* At Step 43, under the instruction “Fly forward to the roadblock and turn back to the bridge” with the current subgoal “Turn back to the bridge”, the hierarchical memory encodes a concise, struc-

**Instruction:** Go across the bridge ahead. Turn left to face the road, go ahead until reaching a bridge on the left. Turn left and cross the bridge. Fly forward to the roadblock and turn back to the bridge. Cross the bridge and take a slight left to face a blue house and land in front of it and stop.

### Hierarchical Memory

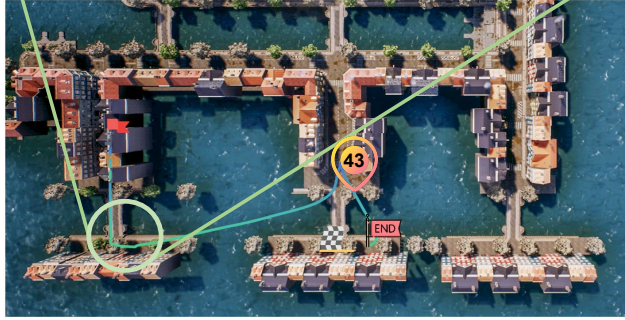


#### Subgoal Judger

**Subgoal\_achieved:** true

The <Current Observation> indicates that the bridge is now in the foreground, approximately 5-10 meters away, which suggests that the drone has reached the bridge. The <Subgoal Memory> shows consistent movement towards the bridge, and the final position aligns with the expected state of having approached the bridge. The <Reference Imagination> is not fully matched...

Concise and effective memory, triggers correct subgoal switching.



43 Current Instruction: Fly forward to the roadblock and turn back to the bridge.  
Current Subgoal: Turn back to the bridge

#### The content of Hierarchical Memory at Step 43:

##### For <Fly forward to the roadblock>:

I see a cobblestone street with buildings, a blue house 20 meters ahead, and water on the right; I move forward 5 meters.

##### For <Turn back to the bridge>:

At step-1: I see a blue house 10 meters ahead and a canal with a light gray building 5 meters to my right; I turn right 15 degrees.

At step-2: I see a narrow street with buildings on both sides, a large arched doorway 5 meters ahead, and a blue building 5-7 meters to the left; I turn right 15 degrees.

At step-3: I see a narrow street with cobblestones, a light gray building 5 meters ahead, a blue house 3 meters left, and water 10 meters right; I turn right 15 degrees.

### Plain History Buffer

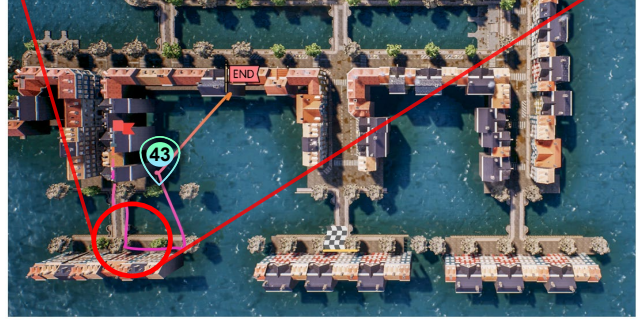


#### Subgoal Judger

**Subgoal\_achieved:** false

The <Current Observation> does not match the <Reference Imagination> or the final state of the <Current Subgoal>...The <Subgoal Memory> indicates that the drone has moved forward and turned, but it has not yet reached a position where the bridge is visible, as required by the <Current Subgoal>.

Too much redundant information in memory, leads to wrong state judgment.



43 Current Instruction: Fly forward to the roadblock and turn back to the bridge.  
Current Subgoal: Fly forward to the roadblock

#### The content of Plain History Buffer at Step 43:

At step-1: I see a cobblestone street with buildings and a speed limit sign; I move forward 5 meters.

At step-2: I see a road ahead with a pedestrian crossing and a bridge 50-70 meters away; I move forward 5 meters.

At step-3: I see a pedestrian crossing and a bridge 50 meters away; I move forward 5 meters.

.....

At step-40: I see a bridge ahead and trees on the left; I move forward 5 meters.

At step-41: I see a bridge 20-30 meters ahead and buildings with red brick facades on the left; I move forward 5 meters.

At step-42: I see a body of water in front, trees and buildings on the left, and a large red-roofed building on the right; I move forward 5 meters.

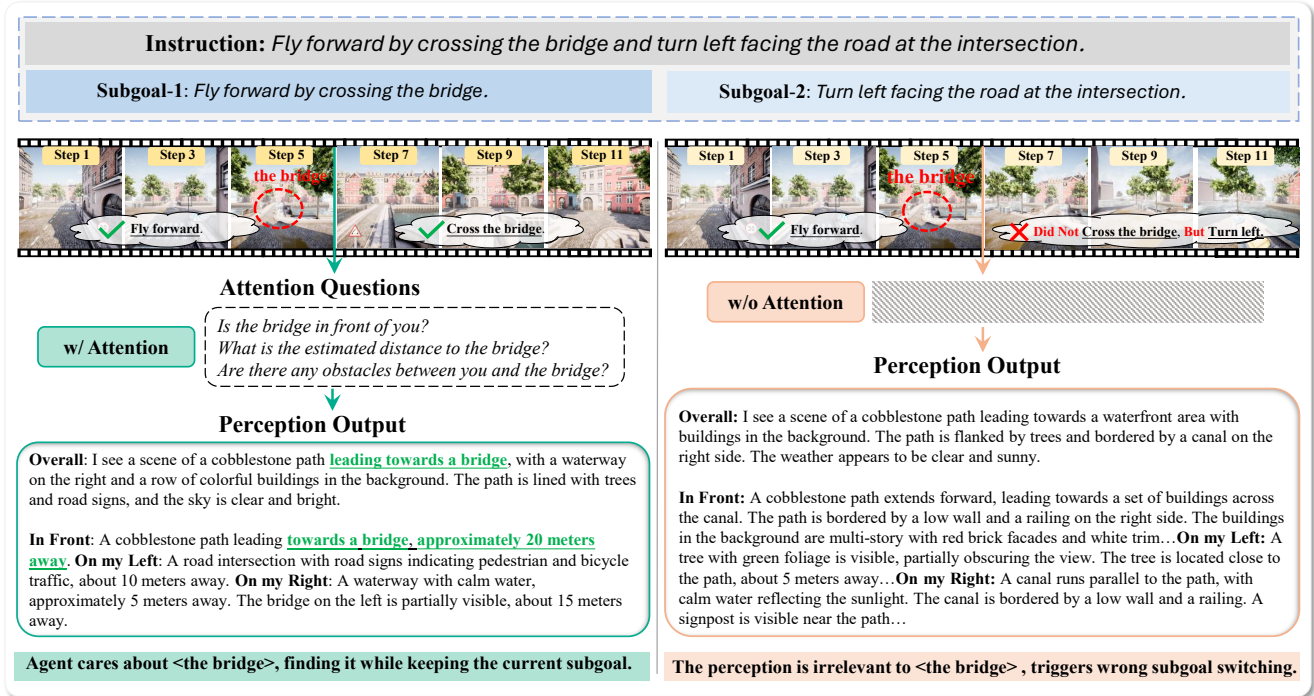
At step-43: I see a canal in front with a small tree on the left, buildings behind the tree, and a bridge in the distance on the right; I move forward 5 meters.

Figure 5. **Comparison of hierarchical memory and plain history buffer** in a complex navigation scenario. The top panels illustrate the impact on subgoal switching: hierarchical memory enables timely and accurate transitions, while the flat buffer leads to delayed or incorrect switching. The bottom panels compare memory content at Step 43 under the same subgoal, in which Hierarchical memory provides a concise, structured summary of key events, whereas the plain buffer stores a lengthy, unstructured sequence, leading to information overload and ambiguous decisions.

structured summary of key events and subgoal states. This abstraction supports precise state judgment and efficient instruction execution. Conversely, the plain history buffer accumulates all 43 low-level observations and actions in an unorganized list, resulting in information overload and ambiguous decision-making.

To summarize, Hierarchical memory provides compact and structured representations essential for subgoal tracking and long-horizon planning. Its absence leads to redundant internal states and degraded navigation performance.

✎ **Analysis of Attention.** Fig. 6 compares the agent's behavior with and without the Attention module. To elaborate,



when executing the instruction “*Fly forward by crossing the bridge and turn left facing the road at the intersection*”, the behaviors diverge clearly:

✗ *Without Attention:* Without the attention module, the agent’s perception becomes scattered and shifts away from the bridge. As a result, the agent ignores the bridge and wrongly extracts subgoals from the instruction, failing to cross the bridge and making incorrect decisions.

### 3. Human Study Details

### 3.1. Design of the Human Study Process

➡ **Introduction Page:**

① **Task definition:** “UAV VLN (*Drone Vision-Language Navigation*) requires drones to autonomously navigate in a 3D continuous space from a first-person perspective based on natural language instructions.”;

③ **Scoring Criteria:** “5 being the best, 1 being the worst, with only one video allowed to receive a score of 5.”, and “Reference scoring criteria: whether it matches the instructions, whether it resembles human pilot exploration behavior, whether there are critical issues like collisions, and whether it flies out of the scene for extended periods.”;

④ **Brief Introduction of Trajectories in the Questionnaire:** “All videos reflect the actual performance of the methods. If issues such as short video length, flying out of the scene, collisions, or clipping occur, these are not due to webpage loading or rendering problems.”;

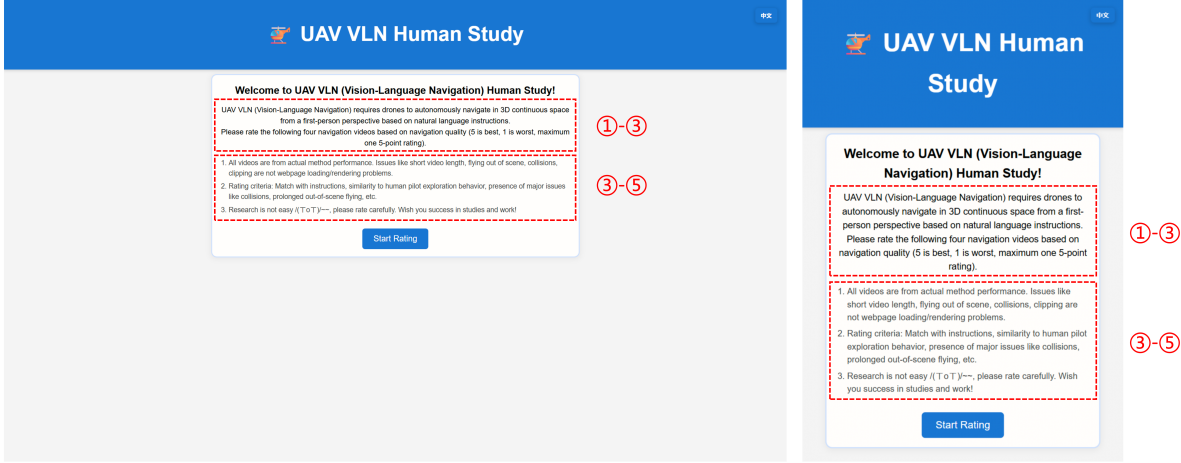


Figure 7. Introduction page of our questionnaire.

⑤ **Thank You Note to Participants.** “Research is challenging/(ToT)/, so we hope everyone will rate carefully. We wish you academic success and a smooth career ahead!”.

#### ► Evaluation Page:

As Fig. 8, the evaluation page also consists of three parts:

① **Task Definition**, which is the same as Introduction Page;

② **Current Progress and the Current Instruction**, e.g. “*Review Progress: 1/10. Instruction: Turn right facing the river and follow the water way until reaching a bridge. Fly forward along the river, pass two bridges, and then turn left towards the road at the first intersection on the left side of the riverbank. Go forward to the end of the road and stop in front of the red building.*”;

③ **Video Display and Scoring.** Four embedded video players (Ours, BaseModel, NavGPT, and DiscussNav) are presented, each followed by a scoring interface that allows participants to assign a score from 1 (worst) to 5 (best), with the constraint that only one video per group may receive a score of 5.

#### ► Farewell Page:

As shown in Fig. 9, the farewell page once again expresses our sincere gratitude to all participants for their time and contributions to the study. “*Review Completed. Thank You for Your Participation! You have finished reviewing all tasks. Thank you for your valuable time and feedback!*”

### 3.2. Analysis of Human Study Results

Our human study aimed to compare the performance of our proposed FineCog-Nav against three baseline methods: BaseModel (a prompt based on the STMR [2] action maker, covering basic instruction understanding, memory, and planning), NavGPT, and DiscussNav. A total of 69 participants contributed ratings across 10 trajectories per

method, yielding 447 valid rating instances for analysis after removing duplicate responses.

① **Performance Metrics:** The analysis of mean ratings showed that our method achieved the highest score at  $3.474 \pm 1.192$ , significantly outperforming BaseModel ( $2.488 \pm 1.298$ ), DiscussNav ( $2.179 \pm 1.247$ ), and NavGPT ( $1.915 \pm 1.245$ ). Notably, our method received the highest proportion of top ratings—24.2% of its ratings were 5 points—and a majority (52.4%) were rated 4 or 5. In contrast, NavGPT received 58.8% of its ratings as 1 point, reflecting poor perceived performance. This trend is further supported by the rating distribution box plot (Fig. 10(a)), where our method exhibits a higher median (4.0) and substantially fewer low ratings compared to all baselines.

② **Significance Testing:** Pairwise comparisons using the Wilcoxon signed-rank test revealed statistically significant differences between all method pairs (all p-values  $< 0.0001$ ). Effect size analysis confirmed the practical relevance of these differences, with large effect sizes observed when comparing our method against BaseModel (Cohen’s  $d = -0.792$ ), NavGPT ( $d = -1.280$ ), and DiscussNav ( $d = 1.062$ ).

③ **Inter-method Correlation Analysis:** Correlation analysis indicated that participants’ ratings for the baseline methods were moderately to strongly correlated (e.g., BaseModel vs. DiscussNav:  $r = 0.517$ ; NavGPT vs. DiscussNav:  $r = 0.663$ ), suggesting consistent perception patterns among them. In contrast, our method showed very weak correlations with all baselines—BaseModel ( $r = 0.057$ ), NavGPT ( $r = 0.132$ ), and DiscussNav ( $r = 0.175$ )—highlighting its distinct and superior perceived behavior.

④ **Participant Rating Consistency:** Despite notable inter-participant variability in rating tendencies—as illustrated in the participant ratings heatmap (Fig. 10(c))—our

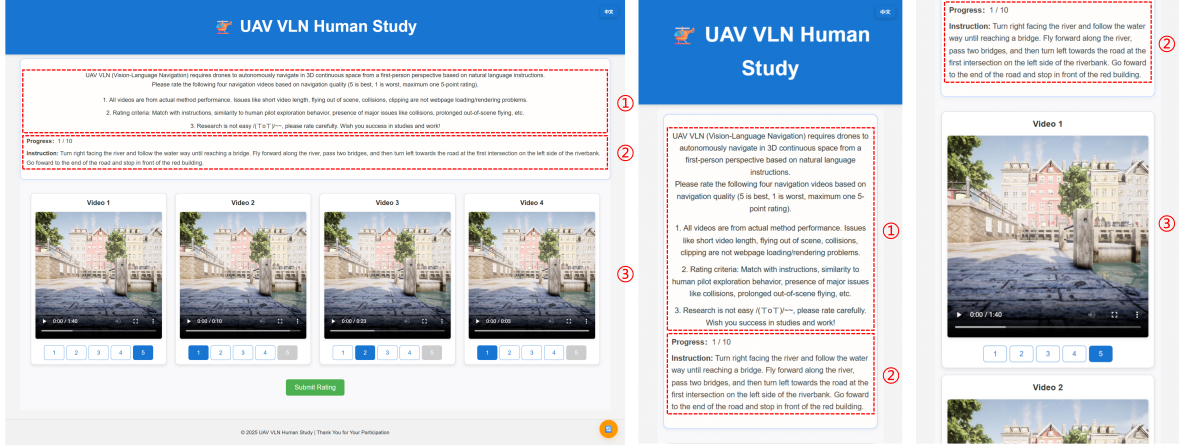


Figure 8. Evaluation page of our questionnaire.

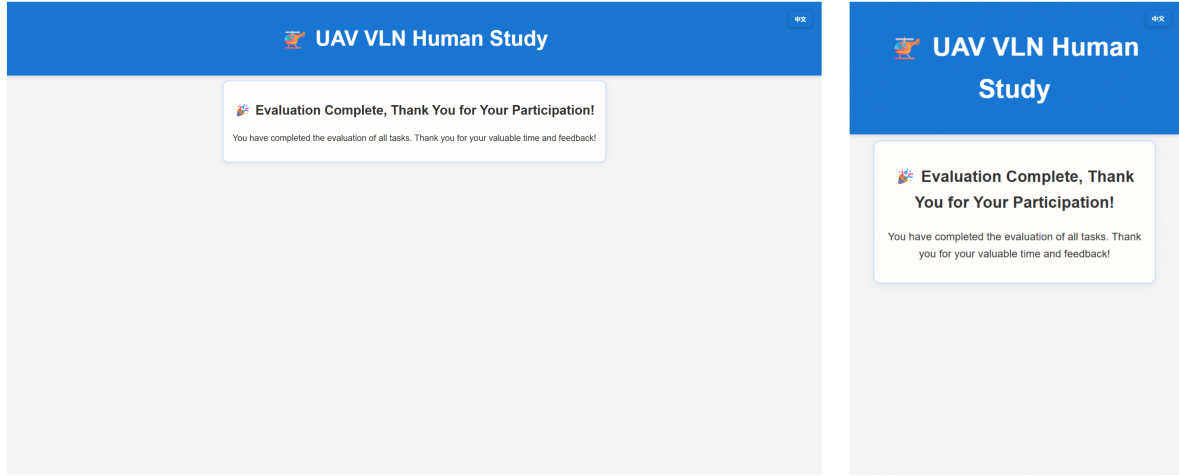


Figure 9. Farewell page of our questionnaire.

method consistently received a higher fraction of favorable ratings (4s and 5s) across nearly all participants compared to the baselines.

Overall, the findings from this human study, comprehensively presented in Fig. 10, provide robust evidence for the effectiveness of our proposed FineCog-Nav over existing baselines.

## 4. More Evaluations

### 4.1. Controlled Evaluation

See, Point, Fly (SPF)[5] is a zero-shot UAV-VLN framework, which shows high success rates in their paper. However, their evaluation is limited to self-constructed tasks that are nearly saturated (success rates of 90–100%), offering little room for meaningful comparison. In contrast, our benchmark encompasses a broader spectrum of scenarios, ranging from relatively easy instances to challenging cases.

To ensure a fair comparison, we curate a subset of our dataset—AerialVLN-Fine-Moderate—that closely aligns with the task design of SPF. The statistics of AerialVLN-Fine-Moderate and the SPF benchmark are presented in Fig. 11. As shown in Tab. 4, on this comparable subset, our method achieves a success rate (SR) of 53.8%, significantly outperforming SPF’s 30.8%, thereby demonstrating that the lower SR observed in our full benchmark primarily stems from increased task difficulty rather than model limitations.

### 4.2. Generalization Evaluation

To further demonstrate the generalization capability of our method across diverse benchmarks, we evaluate it on the OpenFly dataset[3], specifically using the AirSim16 scene, the results are shown in Tab. 5. Our approach matches the performance of the trained one and surpasses SPF [5](the zero-shot UAV-VLN baseline), highlighting its generalization ability.

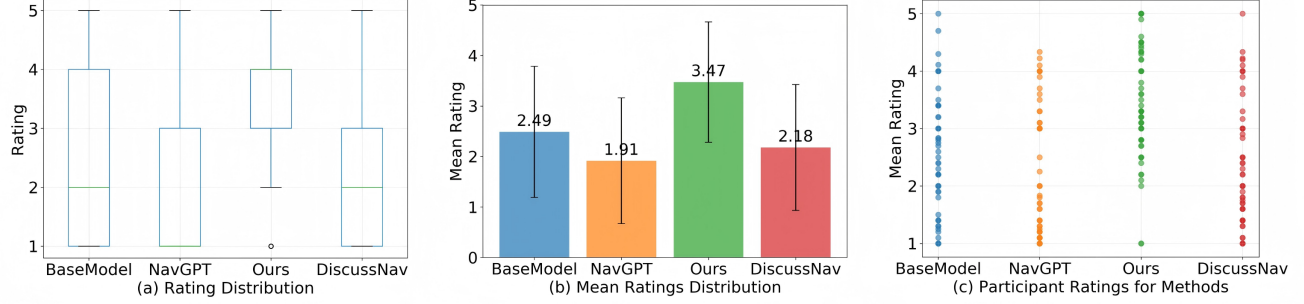


Figure 10. Human study results analysis: (a) Rating Distribution by Method (Box Plot), (b) Mean Ratings by Method (with Standard Deviation), (c) Participant Ratings for Different Methods.

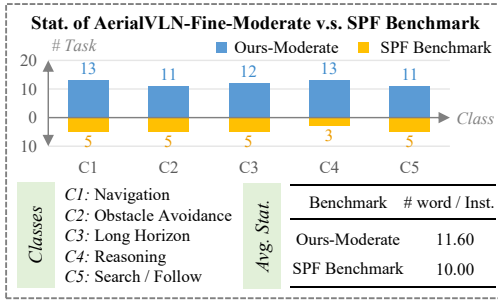


Figure 11. Statistics of AerialVLN-Fine-Moderate

Table 4. Performance on AerialVLN-Fine-Moderate

Method	C1	C2	C3	C4	C5	All
See,Point,Fly	30.8	9.1	16.7	15.4	9.1	16.7
BaseModel	7.7	9.1	8.3	7.7	9.1	8.3
DiscussNav	7.7	9.1	25.0	23.1	18.2	16.7
NavGPT	15.4	0.0	16.7	7.7	0.0	8.3
FineCog-Nav	53.8	18.2	33.3	15.4	18.2	28.3

Table 5. Performance on OpenFly(AirSim 16 subset)

Method	Trained	SR/%	OSR/%	SPL/%	NE/m
OpenFly	✓	10.6	57.9	8.3	178.9
See,Point,Fly	✗	3.5	5.4	2.6	88.2
FineCog-Nav	✗	9.9	32.7	9.1	80.8

### 4.3. Efficiency Evaluation

We compare the computational efficiency of FineCog-Nav with multiple baselines on AerialVLN-Fine (Tab. 6). We report the average amount of API calls, token consumption per step, and latency per step.

While achieving the highest success rate (6.0%), FineCog-Nav maintains a favorable efficiency-performance

Table 6. Cost Analysis of FineCog-Nav and Baselines on AerialVLN-Fine.

Method	SR/%	Calls	Tokens	Latency/s
BaseModel	2.7	2.0	2.6K	18.8
NavGPT	0.0	3.1	10.7K	116.4
DiscussNav	2.7	6.2	18.7K	80.5
FineCog-Nav	6.0	4.9	3.9K	23.5

Table 7. Ablation Analysis of Collision Warning Module on AerialVLN-Fine.

Method	w/ collision warning				w/o collision warning			
	SR/%	OSR/%	NE/m	nDTW/%	SR/%	OSR/%	NE/m	nDTW/%
BaseModel	2.7	6.3	270.1	10.7	1.0	3.7	317.1	13.4
NavGPT	0.0	0.7	110.9	15.9	0.3	0.7	122.4	14.3
DiscussNav	2.7	3.3	98.4	19.6	1.7	1.7	111.1	17.3
FineCog-Nav	6.0	9.0	91.4	22.8	2.7	5.3	100.4	19.2

balance. Compared with two framework-based methods (DiscussNav and NavGPT), FineCog-Nav significantly reduces computational overhead, e.g., token usage (3.9K v.s. 10.7K/18.7K) and latency (23.5s v.s. 116.4s/80.5s), while achieving better performance. Compared with the BaseModel, although incurring modest additional cost, FineCog-Nav improves the success rate by more than 2 $\times$ .

Overall, FineCog-Nav achieves a better balance between reasoning capability and computational efficiency.

### 4.4. Ablation of Collision Warning

We evaluated all methods w/ and w/o the collision warning ( $W_t$ ) to quantify its impact, shown in Tab. 7. Removing the collision warning module leads to consistent performance drops across methods, especially in SR and OSR, indicating that it effectively reduces collision-induced failures and improves navigation success. At the same time, FineCog-Nav outperforms all baselines in both settings, with consistent relative gains. This suggests that the improvements of our method do not rely on this module, but stem from its stronger reasoning and decision-making capabilities.

## 5. Detailed Prompts for BaseModel

In designing the Basemodel, we retained only the most basic and minimal prompt to minimize performance gains from prompt engineering. Specifically, the Basemodel consists of two modules: Perception, responsible for scene understanding, and Action, responsible for decision-making.

### 5.1. Perception

#### Perception

Describe the image.

We present a demonstration of the perception output generated by Qwen2.5-VL-32B-Instruct for an arbitrary input image.

#### Perception

##### Input Information



Describe the image.

##### Output Information

The image depicts a picturesque urban scene with a European architectural style. Here are the key details:

##### ### \*\*Foreground:\*\*

- The foreground shows a portion of a roof, likely from a building or structure in the viewer's immediate vicinity. The roof is made of dark, textured tiles, suggesting a traditional or historical design.

##### ### \*\*Middle Ground:\*\*

- The middle ground features a street that runs diagonally through the scene, leading the viewer's eye toward the background. The street is relatively empty, with no visible vehicles or people.
- On either side of the street, there are rows of buildings with classic European architecture. The buildings are primarily made of red brick, with

#### Perception

- pitched roofs and chimneys, giving them a uniform and historic appearance.
- The buildings have multiple stories, with large windows and symmetrical facades. Some of the buildings have ornate details, such as decorative trim or small balconies, adding to their charm.
- Trees are planted along the sidewalks, providing greenery and shade. The trees are lush and well-maintained, contributing to the overall aesthetic of the scene.

##### ### \*\*Background:\*\*

- The background shows a clear blue sky with scattered white clouds, indicating a bright and sunny day.
- In the distance, the horizon is visible, suggesting that the city is near a body of water or an open landscape, though the specific details are not clear.

##### ### \*\*Overall Atmosphere:\*\*

- The image conveys a calm and serene urban environment, with a focus on historical architecture and well-maintained greenery. The lighting is bright and natural, enhancing the colors and details of the scene.
- The perspective and composition create a sense of depth, drawing the viewer into the street and the buildings beyond.

This image likely represents a digitally rendered or computer-generated scene, given the clean lines, lighting, and overall aesthetic quality. It could be from a video game, architectural visualization, or a digital art project.

#### 5.1.1. Rule-based Collision Estimator

The rule-based collision estimator is shared by all baselines and FineCog-Nav. Considering that collisions are rare during TURN LEFT or TURN RIGHT, and that MOVE LEFT and MOVE RIGHT are also infrequently used, we estimate collision risk only for MOVE FORWARD, ASCENT, and DESCENT.

To estimate the risk of MOVE FORWARD, we examine the central region of the depth map. Because the camera's field of view is limited, depth values above and below the image are not directly observable. Estimating the risk of ASCENT and DESCENT therefore requires an additional assumption. By treating the unseen upper and lower regions as planar surfaces, we infer their distances using the vertical height gradient within the visible area.

```

# Check whether there is collision risk
in performing the action
def _check_collision(self, depth_img,
    action, img_width=672, img_height
    =672, drone_width=1.0, drone_height
    =0.1, fov=90, distance=5.1):
    # depth_img.shape # (h, w, c)
    pixel_angle = fov / img_width
    # the coordinates of center pixel
    center_x = img_width // 2
    center_y = img_height // 2
    # angle of deviation of the edge of
    the central region from center
    half_angle_x=np.arctan(drone_width/(2*
        distance))*(180/np.pi)
    half_angle_y=np.arctan(drone_height
        /(2*distance))*(180/np.pi)
    # size of the central region
    half_width = math.ceil(half_angle_x /
        pixel_angle)
    half_height = math.ceil(half_angle_y /
        pixel_angle)
    if action == 1: # MOVE_FORWARD
        for dx in range(-half_width,
            half_width):
            for dy in range(-half_height,
                half_height):
                x = center_x + dx
                y = center_y + dy
                if x<0 or x>=img_width or y<0 or
                    y>=img_height:
                    continue
                if depth_img[y, x] < distance:
                    # depth less than threshold ->
                    risky
                return True
            return False
    elif action == 4: # ASCENT
        # Calculate the height of each pixel
        height_map=np.zeros_like(depth_img)
        for y in range(img_height):
            angle_y_tan=np.tan(abs(y-center_y)*
                pixel_angle*(np.pi/180))
            height_map[y]=angle_y_tan*depth_img
                [y]
        height=math.ceil(img_height*0.05)
        # Calculate the gradient in the
        vertical direction
        gradient_y = np.gradient(height_map,
            axis=0)
        gradient_threshold = 0.02
        for dx in range(-half_width,

```

```

        half_width):
            x = center_x + dx
            for dy in range(0, height):
                y = img_height + dy
                if x<0 or x>=img_width or y<0 or
                    y>=img_height:
                    continue
                gradient = abs(gradient_y[y, x])
                if height_map[y,x]<distance and
                    gradient<=gradient_threshold:
                    # height and gradient less than
                    threshold -> risky
                return True
            return False
        elif action == 5: # DESCENT
            # Same as ASCENT
        else:
            return False
    def check_collision(self, depth_img):
        risks = ""
        if _check_collision(depth_img, 1):
            risks += "MOVE_FORWARD_will_collide_
                with_objects._"
        if _check_collision(depth_img, 4,
            distance=2.2):
            risks += "ASCEND_will_collide_with_
                objects._"
        if _check_collision(depth_img, 5,
            distance=2.2):
            risks += "DESCEND_will_collide_with_
                objects._"
        if len(risks) == 0:
            risks = "None"
        return risks

```

## 5.2. Action

The Action module, following current modular design trends, incorporates fundamental components for instruction understanding, memory management, and long-term planning, operating with standardized input-output formats.

### Action

#### [Task Description]

You are an embodied unmanned aerial vehicle that navigates in the real world. You need to explore between some places marked and ultimately find the destination to stop.

#### [Input Format]

'Instruction' is a global, step-by-step detailed guidance.

## Action

‘Observation’: the description of the input image.  
‘History’ is your previously executed actions.  
‘Plan’ records previous long-term multi-step planning info that you can refer to now.  
‘Collision Warning’ is the tips to avoid collisions and infer your relative position with surroundings.

### [Output Format]

```
““json
{{
  “Thought”: “your thoughts about the task, which
may include your comprehension, surroundings,
history, and etc.”,
  “Plan”: “your updated plan.”,
  “Action_Number”: “only your next one action
number.”
}}
””
```

You must follow the Output Format to output, and you must not output any other words. Output with plain text format. Think step by step. First, judge by the input image, give a first ‘Thought’, and depict your orientation. Second, check that if a landmark in the current ‘plan’ is within 5 meters of your current position, then based on ‘Instruction’ and the previous ‘Plan’, update your multi-step ‘Plan’. Each plan needs to follow a state word (TODO, In Process, Completed). Finally, judge by the input image again, and give the only one specific ‘action number’ according to the [Valid Action List].

### [Valid Action List]

- 0: TASK\_FINISH
- 1: MOVE\_FORWARD (5 meters)
- 2: TURN\_LEFT (15 degrees)
- 3: TURN\_RIGHT (15 degrees)
- 4: ASCENT (2 meters)
- 5: DESCENT (2 meters)
- 6: MOVE\_LEFT (5 meters)
- 7: MOVE\_RIGHT (5 meters)

### [Input]

Instruction: {instruction}  
Observation: {observation}  
History: {history}  
Plan: {plan}  
Collision Warning: {warning}

Here, we further demonstrate how the action module processes the output from the perception module. After obtaining a scene description from the perception module, the action module parses this information, integrates it with the current navigation context (e.g., instruction history, UAV state), and performs reasoning to determine the next executable action. We use Qwen2.5-72B-Instruct [8] as the LLM.

## Action

### Input Information

#### [Input]

Instruction: Reduce to the second floor height, then turn right to face the street, and then proceed straight ahead to the brown wooden door of the light red building at the corner of the road. Turn your back and face the road, then fly straight forward to the intersection beside the bridge. Turn right, face the white building at the end of the bridge and the road, and then fly straight along the road. Turn right and face the tree at the end of the road, then fly forward to the road intersection. Turn right and head towards the sidewalk on the left side of the building with the dog mural. Walk forward to the building with the dog mural, then turn left to face the red building with the green door on the other side of the river, and then fly over the river, then turn right to face the building with the green roof and red walls, then fly forward to the red building and land there.

Observation: The image depicts a picturesque urban scene with a European architectural style. Here are the key details:

#### ### \*\*Foreground:\*\*

- The foreground shows a portion of a roof, likely from a building or structure in the viewer’s immediate vicinity. The roof is made of dark, textured tiles, suggesting a traditional or historical design.

#### ### \*\*Middle Ground:\*\*

- The middle ground features a street that runs diagonally through the scene, leading the viewer’s eye toward the background. The street is relatively empty, with no visible vehicles or people.
- On either side of the street, there are rows of buildings with classic European architecture. The buildings are primarily made of red brick, with pitched roofs and chimneys, giving them a uniform and historic appearance.

### Action

- The buildings have multiple stories, with large windows and symmetrical facades. Some of the buildings have ornate details, such as decorative trim or small balconies, adding to their charm.
- Trees are planted along the sidewalks, providing greenery and shade. The trees are lush and well-maintained, contributing to the overall aesthetic of the scene.

#### ### \*\*Background:\*\*

- The background shows a clear blue sky with scattered white clouds, indicating a bright and sunny day.
- In the distance, the horizon is visible, suggesting that the city is near a body of water or an open landscape, though the specific details are not clear.

#### ### \*\*Overall Atmosphere:\*\*

- The image conveys a calm and serene urban environment, with a focus on historical architecture and well-maintained greenery. The lighting is bright and natural, enhancing the colors and details of the scene.
- The perspective and composition create a sense of depth, drawing the viewer into the street and the buildings beyond.

This image likely represents a digitally rendered or computer-generated scene, given the clean lines, lighting, and overall aesthetic quality. It could be from a video game, architectural visualization, or a digital art project.

History: ['DESCENT (2 meters)', 'ASCENT (2 meters)']

Plan: Reduce to the second floor height (In Process), Turn right to face the street (TODO), Proceed straight ahead to the brown wooden door of the light red building at the corner of the road (TODO), Turn back and face the road (TODO), Fly straight forward to the intersection beside the bridge (TODO), Turn right and face the white building at the end of the bridge and the road (TODO), Fly straight along the road (TODO), Turn right and face the tree at the end of the road (TODO), Fly forward to the road intersection (TODO), Turn right and head towards the sidewalk on the left side of the building with the dog mural (TODO), Walk forward to the building with the dog mural (TODO), Turn left to face the red building with the green door on the other side of the

### Action

river (TODO), Fly over the river (TODO), Turn right to face the building with the green roof and red walls (TODO), Fly forward to the red building and land there (TODO).

Collision Warning: None

#### Output Information

```
““json
{
  “Thought”: “I am currently at the height of the second floor, as I have just completed an ascent and descent. The image shows a European-style urban scene with a street running diagonally through the middle ground. The street is lined with red brick buildings and trees. I need to turn right to face the street and proceed to the brown wooden door of the light red building at the corner of the road.”,
  “Plan”: “Reduce to the second floor height (Completed), Turn right to face the street (In Process), Proceed straight ahead to the brown wooden door of the light red building at the corner of the road (TODO), Turn back and face the road (TODO), Fly straight forward to the intersection beside the bridge (TODO), Turn right and face the white building at the end of the bridge and the road (TODO), Fly straight along the road (TODO), Turn right and face the tree at the end of the road (TODO), Fly forward to the road intersection (TODO), Turn right and head towards the sidewalk on the left side of the building with the dog mural (TODO), Walk forward to the building with the dog mural (TODO), Turn left to face the red building with the green door on the other side of the river (TODO), Fly over the river (TODO), Turn right to face the building with the green roof and red walls (TODO), Fly forward to the red building and land there (TODO).”,
  “Action_Number”: 3
}
““
```

## 6. Detailed Prompts for Framework Baselines

Since most existing zero-shot methods are designed for discrete ground-level navigation, we made specific adjustments to each method to adapt them to the aerial VLN setting. These modifications mainly include:

1. Replacing all original outputs designed to select a *way-point* with *action* selection from a predefined set of UAV-executable commands, including ascend, descend, turn left, turn right, move left, and move right.
2. Replacing all terms like *indoor* and *vehicle* in the prompts with *outdoor* and *UAV*, respectively.

The modified versions of the baseline codes will be released as open-source. We provide the specific modifications made for each baseline method. All deleted content from the original prompts is highlighted in light gray, while the added modifications are marked in red. The details are as follows.

## 6.1. NavGPT

NavGPT[10] originally uses BLIP-2 [6] and Faster R-CNN [4] to generate scene descriptions, GPT-4 [1] for navigation reasoning, and GPT-3.5-turbo for history management. In our adaptation, we replace GPT-4 with the LLM defined in our setup, BLIP-2 with our selected VLM, and convert its discrete waypoint output to UAV-compatible action outputs. We want to highlight that NavGPT does not use standalone prompts to obtain scene information; instead, it relies on pre-generated JSON files containing visual observations of the environment. To align it with our unified prompt-based framework, We adopt a basic scene understanding module, which offers basic yet clearer detail description compared to the original JSON format. We randomly select one scene from the original R2R Json file and our AerialVLN-Fine dataset to illustrate the prompt here.

### Original JSON File

A series of hallways, rooms, and bathrooms with wooden floors, white walls, and various light fixtures and furniture.

### Generated Scene Description

The image depicts an outdoor scene with a relatively simple and open layout. Here is a detailed description of the scene:

### \*\*Objects and Layout:\*\*

1. \*\*Foreground:\*\*

- The ground is a flat, smooth, concrete surface, likely a rooftop or a paved area.
- There is a small, cylindrical object (possibly a barrel or a container) placed near the center-right of the foreground. It is dark in color and appears to be made of metal or a similar material.

2. \*\*Left Side:\*\*

- A tall, red brick wall runs vertically along the left side of the scene. The wall is solid and appears to be part of a building or structure.
- The wall extends from the foreground into the background, creating a boundary on the left side of the scene.

3. \*\*Right Side:\*\*

- On the right side, there is a partial view of another structure or wall, which appears to be made of a different material (possibly metal or concrete) and is

painted in a dark color.

- The right side is less prominent compared to the left, and the structure is not fully visible.

4. \*\*Background:\*\*

- Beyond the wall on the left, there is a tree with green foliage. The tree is relatively large and occupies a significant portion of the background on the right side.

- In the far background, there is a power line structure (a tall electrical tower) visible, with multiple power lines running horizontally across the scene. The lines stretch from the left side of the image toward the right.

- The sky is clear with a few scattered clouds, indicating a daytime scene with good visibility.

### \*\*Relative Positions:\*\*

- The cylindrical object is positioned closer to the right side of the scene and slightly in front of the viewer.

- The red brick wall is on the left, running vertically and parallel to the edge of the scene.

- The tree is located behind the wall, slightly to the right of the center.

- The power lines and tower are in the far background, extending horizontally across the scene.

### \*\*Possible Directions for Navigation:\*\*

1. \*\*Forward (Straight Ahead):\*\*

- Moving forward would take you along the concrete surface, away from the viewer's perspective. The path appears clear, but the cylindrical object might be an obstacle if you need to navigate around it.

2. \*\*Left:\*\*

- Moving left would bring you closer to the red brick wall. The wall is a solid barrier, so navigating to the left would require staying close to the wall or finding a way around it.

3. \*\*Right:\*\*

- Moving right would take you toward the dark structure on the right side. The area appears open, but the partial structure might limit movement in that direction.

4. \*\*Backward:\*\*

- Moving backward would take you away from the scene, back toward the origin point of the viewer.

### \*\*Obstacles:\*\*

- The \*\*cylindrical object\*\* in the foreground could be an obstacle if precise navigation is required.

- The \*\*red brick wall\*\* on the left side is a solid barrier that would need to be navigated around if moving left is necessary.

- The \*\*power lines\*\* in the background are high

and do not pose a direct obstacle, but they are a notable feature in the scene.

### \*\*Landmarks:\*\*

- The **red brick wall** is a prominent landmark on the left side, providing a clear boundary.
- The **tree** in the background is a natural landmark, helping to orient the scene.
- The **power lines and tower** are a distinct feature in the background, useful for navigation or orientation.

### \*\*Overall Impression:\*\*

The scene is relatively open and flat, with clear boundaries provided by the wall and the tree. The cylindrical object and the wall are the primary obstacles, while the power lines and tree serve as notable landmarks. The concrete surface suggests a rooftop or an open area, and the overall layout is simple and navigable with some attention to the obstacles.

For details, please refer to the prompt design details.

## NavGPT Prompt

### Instruction Split Prompt

Given the long instruction: {instruction}  
Divide the long instruction into action steps with detailed descriptions in the following format:

Action plan:

1. action\_step\_1
2. action\_step\_2

...

Action plan:

### Get Scene Prompt

Please describe the current outdoor scene as detailed as possible, including the objects you can see, their relative positions, the layout of the scene, and the possible directions that can be navigated. Also, point out any obstacles and landmarks that might be helpful for navigation.

### Action Make Prompt

You are an agent following an action plan to navigation in indoor outdoor environment.

Action plan: {action\_plan}

Action list: ['TASK\_FINISH', 'MOVE\_FORWARD (5 meters)', 'TURN\_LEFT (15 degrees)', 'TURN\_RIGHT (15 degrees)', 'ASCENT (2 meters)', 'DESCENT (2 meters)', 'MOVE\_LEFT (5 meters)', 'MOVE\_RIGHT (5 meters)']

## NavGPT Prompt

You are currently at one of the steps in the plan. You will be given the history of previous steps you have taken, the current observation of the environment, and the navigable viewpoints for the next step **current position**. You should:

- 1) evaluate the history and observation to decide which step of action plan you are at.
- 2) choose one viewpoint from the navigable viewpoints **the next action from the action list**.

Each navigable viewpoint has a unique ID, you should only answer the ID **action** in the Final Answer.

Starting below, you should strictly follow this format:

History: the history of previous steps you have taken

Observation: the current observation of the environment **current position**

Navigable viewpoints: the navigable viewpoints for the next step

Thought: your thought on the next step

Collision Warning: Tips to avoid collisions and infer your relative position with surroundings

Final Answer: 'viewpointID' 'action'

Begin!

History: {history}

Observation: {observation}

Navigable viewpoints: {navigable\_viewpoints}

**Collision Warning: {warning}**

Thought:

### History Manage Prompt

You are an agent navigating in indoor an outdoor environment.

You have reached a new viewpoint **position** after taking the previous action. You will be given the navigation history, the current observation of the environment **current position**, and the previous action you took.

You should:

- 1) Evaluate the new observation and history.
- 2) Update the history with the previous action and the new observation.

History: {history}

Previous action: {previous\_action}

Observation: {observation}

Update history with the new observation:

### Back Trace Prompt

You are an agent following an action plan to navigation in indoor outdoor environment.

## NavGPT Prompt

You are currently at an intermediate step of the trajectory but seems going off the track. You will be given the action plan describing the whole trajectory, the history of previous steps you have taken, the observations of the viewpoints along the trajectory **current position**.

You should evaluate the history, the action plan and the observations along the way to decide the viewpoints **next action** to go back to.

Each navigable viewpoint has a unique ID, you should only answer the ID in the Final Answer.

Action list: ['TASK\_FINISH', 'MOVE\_FORWARD (5 meters)', 'TURN\_LEFT (15 degrees)', 'TURN\_RIGHT (15 degrees)', 'ASCENT (2 meters)', 'DESCENT (2 meters)', 'MOVE\_LEFT (5 meters)', 'MOVE\_RIGHT (5 meters)']

You must choose one from the navigable viewpoints **action list**, DO NOT answer None of the above.

Starting below, you should follow this format:

Action plan: the action plan describing the whole trajectory

History: the history of previous steps you have taken

Observation: the observations of each viewpoint along the trajectory **the current position**

Thought: your thought about the next step

Final Answer: 'viewpointID **action**'

Begin!

Action plan: {action\_plan}

History: {history}

Observation: {observation}

Thought:

## 6.2. DiscussNav

DiscussNav[7] employs BLIP-2 and RAM [9] for visual understanding, and GPT-4 for reasoning across four expert modules. Like NavGPT, DiscussNav was designed for indoor, discrete VLN tasks. We apply the same modifications: replacing LLM/VLM modules and adjusting the input/output formats to suit UAV navigation.

## DiscussNav Prompt

### Instruction Analysis Experts

#### System Prompt:

You are an **action a sub-instructions** decomposition expert. Your task is to detect all **actions sub-instructions** in the given navigation instruction. You need to ensure the integrity of each **action sub-instruction**.

#### User Prompt:

Can you decompose **actions sub-instructions** in the instruction '{instruction}'? **Actions Sub-instructions:**

#### System Prompt:

You are a landmark extraction expert. Your task is to detect all landmarks in the given navigation instruction. You need to ensure the integrity of each landmark.

#### User Prompt:

Can you extract landmarks in the instruction '{actions **sub-instructions**}'? **Landmarks:**

### Vision Perception Experts

Describe this **indoor** scene in details

### Completion Estimation Experts

#### System Prompt:

You are a trajectory summary expert. Your task is to simplify navigation thought process as short and clear as possible.

#### User Prompt:

Given Thought Process '{thought}', Summarization:

#### System Prompt:

You are a completion estimation expert. Your task is to estimate what **actions sub-instructions** in the instruction have been executed based on navigation history and landmarks.

All **actions sub-instructions** in the instruction are given following the temporal order. Your answer includes two parts: 'Thought' and 'Executed **Actions Sub-instructions**'.

In the 'Thought', you must follow procedures to analyze as detailed as possible what **actions sub-instructions** have been executed:

- (1) What given landmarks of **actions sub-instructions** have appeared in the navigation history?
- (2) Analyze the direction change at each step in the navigation history.
- (3) Estimate each **action sub-instruction** in the instruction based on each step in the navigation history to check their completion.

In the 'Executed **Actions Sub-instructions**', you must only write down **actions sub-instructions** that have been executed without other words. You must

## DiscussNav Prompt

strictly refer original actions **sub-instructions** in the given instruction to estimate.

### User Prompt:

Given Navigation History ‘{history\_traj}’ and Landmarks ‘{landmarks}’, estimate what actions **sub-instructions** in instruction ‘actions **sub-instructions**’ have been executed.

### Decision Testing Experts

#### System Prompt:

You are a thought fusion expert. Your task is to fuse given thought processes into one thought. You need to reserve key information related to actions, landmarks, direction changes. You should only answer fused thought without other words.

#### User Prompt:

Can you help me fuse the thoughts leading to the same movement **direction action**? The thoughts are :{multiple\_thoughts}, Fused thought:

#### System Prompt:

You are a decision testing expert. Your task is to evaluate the feasibility of each movement prediction based on thought process and environment. Then, you will make a final decision about **navigation viewpoint ID movement action** without other words.

#### User Prompt:

Can you help me make a final decision **about the next action**? **Please choose only one executable action from the options available.** The Observation: {observation}, Navigation Instruction: {instruction}, {fused\_pred\_thought\_}, Final Decision:

### Planner Prompt(DiscussNav agent)

You are a navigation agent who follows instruction to move in an **indoor outdoor** environment with the least action steps. **you can only choose one action from the given list:**  
{description}

I will give you one instruction and tell you landmarks. I will also give you navigation history and estimation of executed actions for reference.

You can observe current environment by scene descriptions, scene objects and possible existing landmarks in different directions around you.

Each direction contains navigable viewpoints you can move to. Your task is to predict moving to which navigable viewpoint.

Note that environment direction that contains more landmarks mentioned in the instruction is usually the better choice for you.

## DiscussNav Prompt

**The Collision Warning is the tips to avoid collisions and infer your relative position with surroundings.**

If you are required to go up stairs, you need to move to direction with higher position. If you are required to go down stairs, you need to move to direction with lower position.

You are encouraged to move to new viewpoints to explore environment while avoid revisiting accessed viewpoints in non-essential situations.

If you feel struggling to find the landmark or execute the **action sub-instruction**, you can try to execute the subsequent **action sub-instruction** and find the subsequent landmark.

Your answer includes two parts: ‘Thought’ and ‘Prediction’. In the ‘Thought’, you should think as detailed as possible following procedures:

(1) Check whether the latest executed action has been completed by comparing current environment and landmark in the latest executed action.

(2) Determine the action you should execute and landmark you should reach now. If the latest executed action have not been completed, you should continue to execute it. Otherwise, you should execute the next action in the given instruction.

(3) Analyze which direction in the current environment is most suitable to execute the action you decide and explain your reason.

(4) Predict moving to which navigable viewpoint based on your thought process. **Predict prompted** which action to execute based on your thought process.

Then, please make decision on the next **viewpoint action** in the ‘Prediction’. You must only answer next **viewpoint ID action** in the ‘Prediction’ without other words.

Your decision is very important, must make it very carefully.”

## 7. Detailed Prompts for FineCog-Nav

We follow the order of module descriptions in Figure 2 of the main text to present our detailed prompt designs. For clarity, each module is highlighted using the corresponding background color from the pipeline diagram.

### 7.1. Instruction Parser

Instruction Parser decomposes the long, complex instruction into a sequence of concise, actionable instruction sentences. This segmentation ensures that the agent can focus on localized goals rather than handling the entire instruction at once.

## Instruction Parser

[General TASK]: You are an Instruction Manager for UAV Navigation. You have the following two tasks:

### 1. SENTENCE SEGMENTATION

- Split input text into individual sentences using periods as separators
- Preserve original wording including leading conjunctions (e.g., “and...”)
- Maintain original capitalization and spacing

### 2. LANDMARK EXTRACTION

- Identify ALL navigational landmarks (physical objects/locations)
- Capture full noun phrases following prepositions: to/at/near/above/before
- Pay special attention to landmarks after temporal prepositions (e.g., “after the building”)
- Retain modifiers: “small building”, “shop entrance”, etc.

[NOTE]:

- Verify period placement for accurate segmentation
- Strictly detect temporal clauses and adjust execution order
- Include ALL landmarks per sentence (1-3 typical)
- Ensure landmarks are bound to corresponding actions in complex sentences
- NEVER modify original wording in sub-instructions
- Eliminate all JSON syntax errors

[OUTPUT CONSTRAINTS]:

- Strictly formatted as a JSON code block without any explanatory text
- Always use arrays even for single items

[EXPECTED OUTPUT]:

```
“json
[ { {
  “sub-instruction”: “...”,
  “landmark”: [ “LANDMARK1”,... ] } },
{ {
  “sub-instruction”: “...”,
  “landmark”: [ “LANDMARK1”,... ]
} } ]
““
```

### Input ###

```
<Instructions>:
{navigation_instruction}
```

## 7.2. Subgoal Extractor

During each navigation step, Subgoal Extractor identifies the immediate subgoal based on the current instruction sentence and the perceived scene context. This dynamic subgoal extraction bridges high-level instructions and low-level UAV actions, enabling precise, step-wise planning.

### Subgoal Extractor

[General TASK]: You are a UAV Navigation Strategist. Perform means-end analysis to decompose the <current instruction> into sequential subgoals.

[Input]:

```
<Current Instruction>
<Current Observation>
```

[Detailed Task]: SUBGOAL EXTRACTION

- Identify all embedded subgoals/intermediate objectives within the <Current Instruction>
- Extract complete phrases denoting actions/steps (e.g., “turn left at intersection”)
- Sequence subgoals by PHYSICAL EXECUTION ORDER, prioritizing temporal logic over sentence structure (Example: “Do A when B” → [“B”, “A”])
- Replace pronouns with explicit referents (e.g., “... A and ... it” → [“... A”, “... A”])
- Preserve descriptive modifiers/contextual details.
- Cross-reference <Current Observation> to ensure executable subgoal generation

[WARNING]:

- Carefully follow <Current Instruction>, DO NOT involve extra descriptive words before <landmarks>, DO NOT add extra information not mentioned.

[OUTPUT CONSTRAINTS]:

- Strictly formatted as a JSON code block without any explanatory text
- Always use arrays even for single items

[EXPECTED OUTPUT]:

```
“json
[“SUBGOAL1”, “SUBGOAL2”]
““
```

### Input ###

```
<Current Instruction>: {instruction}
```

```
<Current Observation>: {observation}
```

### 7.3. Attention

The Attention Module is tightly coupled with the Perception module and is responsible for generating targeted questions for the perception module based on the current and upcoming instruction sentences.

#### Attention

[TASK]: You are an attention guider for a drone's vision module. Generate questions considering distinct landmarks in [Instructions] to direct visual attention of Perception Module.

[INPUT]:  
⟨Current Instruction⟩  
⟨Next Instruction⟩

[OUTPUT CONSTRAINTS]:  
1. Strictly formatted as a JSON code block without any explanatory text.  
2. The OUTPUT JSON must include the following information:  
A list (array) where each element is an object with the following contents:  
- "landmark": The name of the landmark.  
- "question": A question regarding the landmark's CURRENT status(e.g. position or distance, etc.), guiding the attention of the perception module.

[NOTE]:  
1. Question examples:  
- "Is the ⟨landmark⟩ in front of you / on your left / on your right / above / below?"  
- "Is the ⟨landmark⟩ far / near?"  
- "What are the distinct attributes(color, shape, size, etc.) of the ⟨landmark⟩?"  
2. The questions should be specific to the CURRENT status of the landmark and should not reference any actions or future states.

[EXPECTED OUTPUT]:  
"json  
[ { {  
 "landmark": "landmark 1",  
 "question": "...?"  
} },  
... ]  
"

### Input ###  
⟨Current Instruction⟩:{current\_instruction}

⟨Next Instruction⟩:{next\_instruction}

### 7.4. Perception

The Perception Module generates scene descriptions based on the targeted questions provided by the attention module, serving as the foundational component of scene understanding.

#### Perception

[TASK]: You are an advanced multimodal perception system for a drone that navigates in the real world. Your task is to analyze first-person view(the drone's camera is facing forward) image and generate environmental semantics.

[INPUT]:  
⟨Attention⟩: The landmarks you should focus on

[NOTE]:  
1. ONLY include CURRENT VISIBLE landmarks in the ⟨Attention⟩.

[OUTPUT CONSTRAINTS]:  
1. Strictly formatted as a JSON code block without any explanatory text.  
2. The [OUTPUT] JSON must include the following information:  
- "overall": A string describe the scene according to the image input.  
- "details": A string describe the VISIBLE landmarks in the ⟨Attention⟩, estimate their the location and distance(in meters). In Front: describe the middle part of the image, On my Left: describe the left part of the image, On my Right: describe the right part of the image, etc.

[EXPECTED OUTPUT]:  
"json  
{ {  
 "overall": "Overall: I see a scene of ...",  
 "details": "In Front: ... On my Left: ... On my Right: ... etc."  
} }  
"

### Input ###  
⟨Attention⟩: {suggestion}

### 7.5. Imagination

Based on the current and next subgoals, along with the landmark list, the Imagination Module imagines the anticipated state after completing the current subgoal.

## Imagination

[TASK]: You are a drone performing navigation task, trying to achieve ⟨subgoal⟩. Now please imagine the final VISUAL state when you have achieved the given ⟨subgoal⟩. First-person View (The drone's camera is facing forward).

[INPUT]:  
⟨Current Subgoal⟩  
⟨Landmarks⟩  
⟨Next Subgoal⟩

[NOTE]:  
1. Your imagination should be a very concise sentence ONLY describing the visual state of ⟨Landmarks⟩ after you achieved the ⟨Current Subgoal⟩, including:  
- The location and distance(in meters) of ⟨Landmarks⟩. Prefer SPATIAL terms: LEFT-/RIGHT/CENTER, etc.  
2. Consider the ⟨Next Subgoal⟩ to infer the final state of ⟨Current Subgoal⟩.  
3. You can only see a small window: a little to the left, right, up, and down (about 45° each way).  
- If something ends up underneath you, behind you, or too high/low (outside that window), do NOT mention it.  
- Example (general): After moving above a landmark, it is now below you and out of view.

[OUTPUT CONSTRAINTS]:  
Strictly formatted as a JSON code block without any explanatory text.

[EXPECTED OUTPUT]:  
```json  
{  
 "state": "When I have finished the ⟨subgoal⟩, I might see ...(the location and distance of the ⟨Landmarks⟩)." }  
}

### Input ###  
⟨Current Subgoal⟩: {subgoal}  
  
⟨Landmarks⟩: {landmark}  
  
⟨Next Subgoal⟩: {next\_subgoal}

## 7.6. Subgoal Judger

The Subgoal Judger determines whether the current subgoal has been successfully completed.

## Subgoal Judger

[TASK]: You are a drone navigation analysis expert. Your task is to estimate whether the subgoal has been achieved or not.

[INPUT]:  
⟨Current Subgoal⟩  
⟨Subgoal Memory⟩: What you have ALREADY DONE and OBSERVED for ⟨Current Subgoal⟩.  
⟨Current Observation⟩  
⟨Reference Imagination⟩: The blind imagined reference VISUAL state when you have achieved the ⟨Subgoal⟩.  
⟨Next Subgoal⟩: The next subgoal you are trying to transition to

[NOTE]:  
1. To make your decision, analyze the inputs as follows:  
- Check the ⟨Current Observation⟩, mainly focus on the status of Landmarks, and then decide if it is time to switch from the ⟨Current Subgoal⟩ to ⟨Next Subgoal⟩.  
- Review the ⟨Subgoal Memory⟩ to check whether the actions taken align with the ⟨Current Subgoal⟩.  
- ⟨Reference Imagination⟩ is a blind guess, take it for reference occasionally.  
2. If the subgoal says \*\*turn left/right\*\* without a specified degree or reference, it means a large turn, usually > 45 degrees( > 3 times).  
3. If the ⟨Current Observation⟩ is not matched with the ⟨Reference Imagination⟩, but still make sense as both initial state of the ⟨Next Subgoal⟩ and final state of the ⟨Current Subgoal⟩, then it might also achieved.  
4. If there are multiple(>15) steps in the ⟨Subgoal Memory⟩, you should check whether you have achieved the ⟨Current Subgoal⟩ before the last step.

[ADDITIONAL REQUIREMENTS]:  
1. [Action Compliance Verification] For each subgoal evaluation, you MUST verify:  
- Every action SPECIFIED in the subgoal description has been executed (cross-validate with ⟨Subgoal Memory⟩)

[OUTPUT CONSTRAINTS]:  
1. Strictly formatted as a JSON code block without any explanatory text.  
2. Please also provide a brief explanation involving your evidence from [INPUT].

### Subgoal Judge

```
[EXPECTED OUTPUT]:
““json
{{
  “subgoal”: “⟨subgoal⟩”,
  “achieved”: ⟨true|false⟩,
  “reason”: “...(brief explanation)”
}}
““

### Input ###
⟨Current Subgoal⟩: {subgoal}

⟨Subgoal Memory⟩: {history}

⟨Current Observation⟩: {scene}

⟨Reference Imagination⟩: {state}

⟨Next Subgoal⟩: {next_subgoal}
```

## 7.7. Multi-Level Memory

The Multi-Level Memory module implements a hierarchical memory structure that integrates both short-term and long-term memory. In detail, this module consists of Step Memory, Subgoal Memory, and Instruction Memory. The corresponding prompts for each component are described.

### Multi-Level Memory(Step Memory)

[TASK]: You are a **Memory Manager** for a drone that navigates in the real world. Generate a memory statement using the provided [INPUT] data.

[INPUT]:  
⟨Current Observation⟩: The scene you are **SEEING**  
⟨Current Action⟩: The action you are **DOING**  
⟨Landmarks⟩

[NOTE]:  
1. Keep accurate DEGREE/DISTANCE in ⟨Action⟩  
2. ONLY includes VISIBLE ⟨Landmarks⟩ in the ⟨Current Observation⟩  
3. Do NOT infer or include any planning steps or actions not explicitly mentioned in the input.

[OUTPUT CONSTRAINTS]:  
1. Strictly formatted as a JSON code block without any explanatory text

### Multi-Level Memory(Step Memory)

2. **Word limit**: < 30 words

[EXPECTED OUTPUT]:  
““json  
{{ “step\_memory”: the memory in the form of “I see ...(observation); I ...(action)”. }}

### Input ###

⟨Current Observation⟩: {observation}

⟨Current Action⟩: {action}

⟨Landmarks⟩: {landmarks}

For each subgoal, we introduce a Subgoal Memory that operates during its execution phase.

### Multi-Level Memory(Subgoal Memory)

[TASK]: You are a **Memory Manager** for a drone that navigates in the real world. **Consolidate** ⟨Raw Subgoal Memory⟩ containing a list of actions and observations in temporal order.

[INPUT]:  
⟨Raw Subgoal Memory⟩  
⟨Landmarks⟩:

[NOTE]:  
1. **Merge the Information**:  
- Merge the same actions(note that turn left and turn right are different actions) and accumulate the numerical values.  
- Only merge **consecutive** actions of the same type (for example, consecutive “move forward” steps).  
- Do not merge actions if they are interrupted by a different action type.  
- Preserve the temporal order of actions and observations.  
- Only include the landmarks that are **VISIBLE** in the ⟨Raw Subgoal Memory⟩, and pay special attention to changes in landmarks..  
2. **Output Requirements**:  
- The output memory should be concise, clear, and logically organized.

[OUTPUT CONSTRAINTS]:  
1. Strictly formatted as a JSON code block without any explanatory text

### Multi-Level Memory(Subgoal Memory)

2. **Word limit**: < 40 words

[EXPECTED OUTPUT]:

```
““json
{{ “subgoal_memory”: “I have seen ...; I ...(ac-
tion)”. }}
““

### Input ###
⟨Raw Subgoal Memory⟩: {raw_memory}

⟨Landmarks⟩: {landmarks}
```

The Instruction Memory is simply a collection of completed subgoal records, without any associated prompts.

## 7.8. Decision Making

The Decision Making module serves as the final stage of the entire framework. Its primary function is to determine the UAV’s next action, such as move forward, turn left/right, ascend, descend, or stop.

### Decision-Making

[TASK]: You are an embodied drone that navigates in the real world. Your task is to CHOOSE the most reasonable action to be executed next from the VALID ACTIONS, so that you could achieve the subgoal.

[INPUT]:

```
⟨Instruction Memory⟩: What you have AL-
READY DONE and OBSERVED for the current
instruction.
⟨Current Observation⟩
⟨Collision Warning⟩
⟨Current Instruction⟩
⟨Current Subgoal⟩: The subgoal of the current
instruction you are working on.
⟨Valid Actions⟩
```

[NOTE]:

1. To make your decision, analyze the inputs as follows:
  - Understand the ⟨Current Subgoal⟩, find out the actions aligned with it.
  - Consider the ⟨Collision Warning⟩ to avoid collisions.
2. **Probability Rules**:
  - Output probabilities for **ALL Valid Actions**

### Decision-Making

- Higher probability = stronger preference
3. Carefully check the ⟨Instruction Memory⟩: If over multiple steps you consistently observe scenes that no visible landmarks - immediately select the action 0

[OUTPUT CONSTRAINTS]:

1. Strictly formatted as a JSON code block without any explanatory text.
2. The [OUTPUT] JSON must include the following information:
  - “thought”: why you choose this action, considering the ⟨Current Subgoal⟩, ⟨Current Observation⟩, ⟨Instruction Memory⟩, ⟨Current Instruction⟩ and ⟨Collision Warning⟩.
  - “probabilities”: probability distribution over the valid action list.
  - “selected\_action”: Explicitly select the action with highest probability. Note that only output the action number.

### INPUT

```
⟨Instruction Memory⟩: {memory}

⟨Current Observation⟩: {observation}

⟨Collision Warning⟩: {collisions_warning}

⟨Current Instruction⟩: {current_instruction}

⟨Current Subgoal⟩: {subgoal}

⟨Valid Actions⟩: {actions}
```

## 8. Execution Details

Since the instructions in our AerialVLN-Fine dataset are refined, to more comprehensively and realistically demonstrate the performance of our method, we select trajectory 3570Y55XZYHE60QQZMK98C5MBKWYG9 from the original AerialVLN-S validation set for detailed analysis. In this example, we apply our FineCog-Nav framework, utilizing Qwen2.5-72B-Instruct [8] as the LLM and Qwen2.5-VL-32B [8] as the VLM. Specifically, our method traverses 75 meters and terminates at step 31. We present the complete information for several representative steps, including the inputs and outputs of each module at every stage of execution.

To simplify the description, we omit the repeated content from [TASK] to **###Input###** as detailed in the previous section, and only present the specific inputs and outputs here.

## 8.1. Step 0

### Instruction Parser

#### Input Information

⟨Instructions⟩: turn left and fly over the yellow truck. turn left and take the next left at another truck. land in front of the back tire of the truck. stop on the sidewalk.

#### Output Information

```
[ { "sub-instruction": "turn left and fly over the yellow truck.",  
  "landmark": [ "the yellow truck" ] },  
  { "sub-instruction": "turn left and take the next left at another truck.",  
    "landmark": [ "another truck" ] },  
  { "sub-instruction": "land in front of the back tire of the truck.",  
    "landmark": [ "the back tire of the truck" ] },  
  { "sub-instruction": "stop on the sidewalk.",  
    "landmark": [ "the sidewalk" ] } ]
```

### Attention

#### Input Information

⟨Current Instruction⟩:  
turn left and fly over the yellow truck.  
Landmark: the yellow truck  
⟨Next Instruction⟩:  
turn left and take the next left at another truck.  
Landmark: another truck

#### Output Information

```
[ { 'landmark': 'the yellow truck', 'question': 'Is the yellow truck in front of you or on your left?' },  
  { 'landmark': 'another truck', 'question': 'Can you see another truck in your current field of view?' } ]
```

### Perception

#### Input Information



⟨Attention⟩:  
Is the yellow truck in front of you or on your left?  
Can you see another truck in your current field of view?

### Perception

#### Output Information

Overall: I see a scene of an urban environment with a clear sky, a tall obelisk in the background, and a brick wall in the foreground. The area appears to be a plaza or open space with paved ground.

In Front: A paved plaza with a brick wall running horizontally across the middle of the image. The wall is about 5 meters away. On my Left: A white marble structure, possibly a monument or sculpture, is visible near the left edge of the image, approximately 3 meters away. On my Right: A dark-colored truck is partially visible behind the brick wall, about 10 meters away. Another truck is not visible in the current field of view.

### Subgoal Extractor

#### Input Information

⟨Current Instruction⟩:  
turn left and fly over the yellow truck.  
⟨Current Observation⟩:  
Overall: I see a scene of an urban environment with a clear sky, a tall obelisk in the background, and a brick wall in the foreground. The area appears to be a plaza or open space with paved ground.  
In Front: A paved plaza with a brick wall running horizontally across the middle of the image. The wall is about 5 meters away. On my Left: A white marble structure, possibly a monument or sculpture, is visible near the left edge of the image, approximately 3 meters away. On my Right: A dark-colored truck is partially visible behind the brick wall, about 10 meters away. Another truck is not visible in the current field of view.

#### Output Information

```
[ "turn left", "fly over the yellow truck" ]
```

### Imagination

#### Input Information

⟨Current Subgoal⟩:  
turn left  
⟨Landmarks⟩:  
the yellow truck  
⟨Next Subgoal⟩:  
fly over the yellow truck

## Imagination

### Output Information

When I have finished the turn left, I might see the yellow truck at the CENTER, about 10 meters ahead.

## Subgoal Judger

### Input Information

⟨Current Subgoal⟩:

turn left

⟨Subgoal Memory⟩:

None

⟨Current Observation⟩:

Overall: I see a scene of an urban environment with a clear sky, a tall obelisk in the background, and a brick wall in the foreground. The area appears to be a plaza or open space with paved ground.

In Front: A paved plaza with a brick wall running horizontally across the middle of the image. The wall is about 5 meters away. On my Left: A white marble structure, possibly a monument or sculpture, is visible near the left edge of the image, approximately 3 meters away. On my Right: A dark-colored truck is partially visible behind the brick wall, about 10 meters away. Another truck is not visible in the current field of view.

⟨Reference Imagination⟩:

When I have finished the turn left, I might see the yellow truck at the CENTER, about 10 meters ahead.

⟨Next Subgoal⟩:

fly over the yellow truck

### Output Information

subgoal: turn left

achieved: false

reason: The ⟨Current Observation⟩ does not match the ⟨Reference Imagination⟩. The yellow truck is not visible in the center of the field of view, and the ⟨Subgoal Memory⟩ indicates no actions have been taken to turn left. The current view is still aligned with the initial state before the turn.

## Decision-Making

with a clear sky, a tall obelisk in the background, and a brick wall in the foreground. The area appears to be a plaza or open space with paved ground.

In Front: A paved plaza with a brick wall running horizontally across the middle of the image. The wall is about 5 meters away. On my Left: A white marble structure, possibly a monument or sculpture, is visible near the left edge of the image, approximately 3 meters away. On my Right: A dark-colored truck is partially visible behind the brick wall, about 10 meters away. Another truck is not visible in the current field of view.

⟨Collision Warning⟩:

**\*\*MOVE\_FORWARD will collide with objects. DESCEND will collide with objects. \*\***

⟨Current Instruction⟩:

turn left and fly over the yellow truck.

⟨Current Subgoal⟩:

turn left

⟨Valid Actions⟩:

1: MOVE\_FORWARD (5 meters)

2: TURN\_LEFT (15 degrees)

3: TURN\_RIGHT (15 degrees)

4: ASCEND (2 meters)

5: DESCEND (2 meters)

6: MOVE\_LEFT (5 meters)

7: MOVE\_RIGHT (5 meters)

### Output Information

thought: The current subgoal is to turn left. The current observation shows a brick wall in front and a white marble structure on the left, which is approximately 3 meters away. The collision warning indicates that moving forward or descending will result in a collision. Therefore, the most reasonable action is to turn left to align with the subgoal and avoid any obstacles.

probabilities: {"1": 0.05, "2": 0.85, "3": 0.05, "4": 0.02, "5": 0.01, "6": 0.01, "7": 0.01}

selected\_action": 2

## Decision-Making

### Input Information

⟨Instruction Memory⟩:

For ⟨turn left⟩:

None

⟨Current Observation⟩:

Overall: I see a scene of an urban environment

## Multi-Level Memory(Step Memory)

### Input Information

⟨Current Observation⟩:

Overall: I see a scene of an urban environment with a clear sky, a tall obelisk in the background, and a brick wall in the foreground. The area appears to be a plaza or open space with paved ground.

### Multi-Level Memory(Step Memory)

In Front: A paved plaza with a brick wall running horizontally across the middle of the image. The wall is about 5 meters away. On my Left: A white marble structure, possibly a monument or sculpture, is visible near the left edge of the image, approximately 3 meters away. On my Right: A dark-colored truck is partially visible behind the brick wall, about 10 meters away. Another truck is not visible in the current field of view.

⟨Current Action⟩:

TURN\_LEFT (15 degrees)

⟨Landmarks⟩:

the yellow truck | another truck

#### Output Information

I see a brick wall 5 meters away and a white marble structure 3 meters to my left; I turn left 15 degrees.

### Multi-Level Memory(Subgoal Memory)

At step-1: I see a brick wall 5 meters away and a white marble structure 3 meters to my left; I turn left 15 degrees.

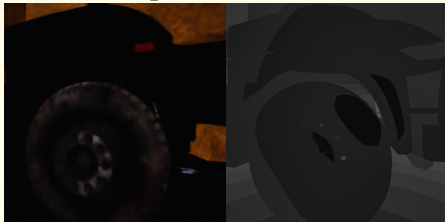
## 8.2. Step 13

To simplify the process, we directly move to Step 13, where a subgoal is completed, to illustrate how the prompt focus shifts before and after the subgoal transition.

Firstly, the Subgoal Judger determines that the current task is complete. Note that the Attention questions and Imagination states have been cached.

### Perception

#### Input Information



⟨Attention⟩:

Is the yellow truck in front of you or on your left?  
Can you see another truck in your current field of view?

### Perception

#### Output Information

Overall: I see a scene with a close-up view of a vehicle's wheel and part of its body, with a dark and somewhat obscured environment.

In Front: The wheel of a vehicle is prominently visible, occupying the central part of the image.

On my Left: The left side of the image shows part of the vehicle's body, which appears to be dark and metallic.

On my Right: The right side of the image shows a portion of the vehicle's body and a small red object, possibly a light or reflector. The yellow truck is not visible in the current field of view.

Another truck is also not visible in the current field of view.

### Subgoal Judger

#### Input Information

⟨Current Subgoal⟩:

fly over the yellow truck

⟨Subgoal Memory⟩:

At step-1: I see a paved area with a bench and a marble-like structure in front, a green dumpster on my left, and a tall obelisk-like structure in the background; I turn left 15 degrees.

At step-2: I see a paved plaza with a bench, a black trash bin, and a tree; I turn left 15 degrees.

At step-3: I see a wide sidewalk, a blue trash bin, a streetlight, a brick wall, a bench, and a marble structure; I turn left 15 degrees.

At step-4: I see a red truck 10 meters left, a brick wall, bench, and trash bin 5 meters right; I turn left 15 degrees.

At step-5: I see a yellow truck 20 meters left, a red truck 15 meters right, and trash bins, bench 5-10 meters right; I move left 5 meters.

At step-6: I see a yellow truck 50 meters ahead, tall buildings on both sides, and a red car 30 meters to the right; I move forward 5 meters.

At step-7: I see a red car 10m right, a yellow truck 30m left, and a blue trash bin 15m right; I turn left 15 degrees.

At step-8: I see a yellow truck 20 meters ahead and a red car 10 meters to the right; I move forward 5 meters.

At step-9: I see a yellow truck 20 meters ahead, tall buildings 30 meters on both sides; I move forward 5 meters.

At step-10: I see a yellow truck 50 meters ahead, a tall building with a 'PARKING' sign 30 meters to

### Subgoal Judger

the left, and another tall building 30 meters to the right; I move forward 5 meters.

At step-11: I see a yellow truck 10 meters ahead and tall buildings on both sides; I move forward 5 meters.

⟨Current Observation⟩:

Overall: I see a scene with a close-up view of a vehicle's wheel and part of its body, with a dark and somewhat obscured environment.

In Front: The wheel of a vehicle is prominently visible, occupying the central part of the image.

On my Left: The left side of the image shows part of the vehicle's body, which appears to be dark and metallic. On my Right: The right side of the image shows a portion of the vehicle's body and a small red object, possibly a light or reflector. The yellow truck is not visible in the current field of view. Another truck is also not visible in the current field of view.

⟨Reference Imagination⟩:

When I have finished the ⟨subgoal⟩, I might see the yellow truck 10 meters below me and out of view.

⟨Next Subgoal⟩:

turn left and take the next left at another truck.

#### Output Information

subgoal: fly over the yellow truck,

achieved: true,

reason: The ⟨Current Observation⟩ shows a close-up view of a vehicle's wheel and part of its body, which is consistent with being directly above the yellow truck. The ⟨Subgoal Memory⟩ indicates that the drone has moved forward and is now very close to the yellow truck, which aligns with the ⟨Reference Imagination⟩ of the yellow truck being 10 meters below and out of view. This suggests that the drone has successfully flown over the yellow truck.

After completing the subgoal, the Subgoal Memory is compressed, and a new empty subgoal memory is initialized. The compressed memory is referred to as raw memory.

### Multi-Level Memory(Subgoal Memory)

#### Input Information

⟨Raw Subgoal Memory⟩:

At step-1: I see a paved area with a bench and a marble-like structure in front, a green dumpster on my left, and a tall obelisk-like structure in the background; I turn left 15 degrees.

### Multi-Level Memory(Subgoal Memory)

At step-2: I see a paved plaza with a bench, a black trash bin, and a tree; I turn left 15 degrees.

At step-3: I see a wide sidewalk, a blue trash bin, a streetlight, a brick wall, a bench, and a marble structure; I turn left 15 degrees.

At step-4: I see a red truck 10 meters left, a brick wall, bench, and trash bin 5 meters right; I turn left 15 degrees.

At step-5: I see a yellow truck 20 meters left, a red truck 15 meters right, and trash bins, bench 5-10 meters right; I move left 5 meters.

At step-6: I see a yellow truck 50 meters ahead, tall buildings on both sides, and a red car 30 meters to the right; I move forward 5 meters.

At step-7: I see a red car 10m right, a yellow truck 30m left, and a blue trash bin 15m right; I turn left 15 degrees.

At step-8: I see a yellow truck 20 meters ahead and a red car 10 meters to the right; I move forward 5 meters.

At step-9: I see a yellow truck 20 meters ahead, tall buildings 30 meters on both sides; I move forward 5 meters.

At step-10: I see a yellow truck 50 meters ahead, a tall building with a 'PARKING' sign 30 meters to the left, and another tall building 30 meters to the right; I move forward 5 meters.

At step-11: I see a yellow truck 10 meters ahead and tall buildings on both sides; I move forward 5 meters.

⟨Landmarks⟩:

the yellow truck | another truck

#### Output Information

I see a paved area, green dumpster, obelisk; I turn left 60 degrees; I see a red truck, yellow truck, trash bins, bench; I move left 5 meters, forward 20 meters; I see a yellow truck, tall buildings, 'PARKING' sign; I move forward 5 meters.

Since this is the final subgoal of the instruction, the Attention questions, Scene description, Subgoal list, and Imagination states must be regenerated.

### Attention

#### Input Information

⟨Current Instruction⟩:

turn left and take the next left at another truck.

Landmark: another truck

⟨Next Instruction⟩:

### Attention

land in front of the back tire of the truck.  
Landmark: the back tire of the truck

#### Output Information

[ {‘landmark’: ‘another truck’, ‘question’: ‘Is the truck on your left?’},  
{‘landmark’: ‘the back tire of the truck’, ‘question’: ‘Is the back tire of the truck in front of you?’} ]

### Subgoal Extractor

#### Input Information

⟨Current Instruction⟩:

turn left and fly over the yellow truck.

⟨Current Observation⟩:

Overall: I see a scene with a close-up view of a vehicle’s wheel and part of its body, with a dark and somewhat obscured environment.

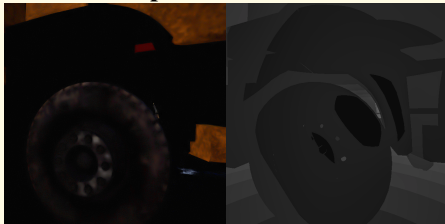
In Front: The wheel of a vehicle is prominently visible, occupying the central part of the image.  
On my Left: The left side of the image shows part of the vehicle’s body, which appears to be dark and metallic.  
On my Right: The right side of the image shows a portion of the vehicle’s body and a small red object, possibly a light or reflector. The yellow truck is not visible in the current field of view. Another truck is also not visible in the current field of view.

#### Output Information

[ “turn left”, “take the next left at another truck” ]

### Perception

#### Input Information



⟨Attention⟩:

Is the truck on your left?

Is the back tire of the truck in front of you?

#### Output Information

Overall: I see a scene with a close-up view of a vehicle’s tire, likely a truck, in a dimly lit environment.

### Perception

In Front: A large tire is prominently visible, occupying the central part of the image. The tire appears to be part of a truck, and its back tire is directly in front of me. Distance: Approximately 1 meter. On my Left: The left side of the image shows part of the truck’s body, which is dark and blends into the background. Distance: Approximately 1-2 meters. On my Right: The right side of the image shows a portion of the truck’s body and a small red light or reflector. Distance: Approximately 1-2 meters.

### Decision-Making

#### Input Information

⟨Instruction Memory⟩:

None

⟨Current Observation⟩:

Overall: I see a scene with a close-up view of a vehicle’s tire, likely a truck, in a dimly lit environment.

In Front: A large tire is prominently visible, occupying the central part of the image. The tire appears to be part of a truck, and its back tire is directly in front of me. Distance: Approximately 1 meter. On my Left: The left side of the image shows part of the truck’s body, which is dark and blends into the background. Distance: Approximately 1-2 meters. On my Right: The right side of the image shows a portion of the truck’s body and a small red light or reflector. Distance: Approximately 1-2 meters.

⟨Collision Warning⟩:

\*\*MOVE\_FORWARD will collide with objects. ASCEND will collide with objects. DESCEND will collide with objects. \*\*  
⟨Current Instruction⟩: turn left and take the next left at another truck.

⟨Current Subgoal⟩:

turn left

⟨Valid Actions⟩:

- 1: MOVE\_FORWARD (5 meters)
- 2: TURN\_LEFT (15 degrees)
- 3: TURN\_RIGHT (15 degrees)
- 4: ASCEND (2 meters)
- 5: DESCEND (2 meters)
- 6: MOVE\_LEFT (5 meters)
- 7: MOVE\_RIGHT (5 meters)

### Decision-Making

#### Output Information

thought: Given the current subgoal to turn left, and considering the collision warnings for moving forward, ascending, and descending, the most reasonable action is to turn left. The current observation shows a large tire and truck body on the left, which aligns with the subgoal of turning left.

probabilities: {"1": 0.05, "2": 0.85, "3": 0.02, "4": 0.01, "5": 0.01, "6": 0.03, "7": 0.01}

selected\_action: 2

### Multi-Level Memory(Step Memory)

#### Input Information

⟨Current Observation⟩:

Overall: I see a scene with a close-up view of a vehicle's tire, likely a truck, in a dimly lit environment.

In Front: A large tire is prominently visible, occupying the central part of the image. The tire appears to be part of a truck, and its back tire is directly in front of me. Distance: Approximately 1 meter. On my Left: The left side of the image shows part of the truck's body, which is dark and blends into the background. Distance: Approximately 1-2 meters. On my Right: The right side of the image shows a portion of the truck's body and a small red light or reflector. Distance: Approximately 1-2 meters.

⟨Current Action⟩:

TURN\_LEFT (15 degrees)

⟨Landmarks⟩:

another truck — the back tire of the truck

#### Output Information

I see a large tire and part of a truck's body; I turn left 15 degrees.

### Multi-Level Memory(Subgoal Memory)

At step-1: I see a large tire and part of a truck's body; I turn left 15 degrees.

## 8.3. Step 14

Our agent finishes a subgoal at step 13. Therefore, at step 14, it starts with a new subgoal and changes in the imagination module.

### Perception

#### Input Information



⟨Attention⟩:

Is the truck on your left?

Is the back tire of the truck in front of you?

#### Output Information

Overall: I see a close-up view of a truck's rear section, including its tire and part of its body.

In Front: The back tire of the truck is directly in front of me, approximately 1 meter away. On my

Left: The side of the truck is visible, extending to the left, with the truck's body occupying most of the left side of the image. On my Right: The truck's rear section, including part of its tailgate and body, is visible, occupying the right side of the image.

The Imagination states are regenerated when needed.

### Imagination

#### Input Information

⟨Current Subgoal⟩:

turn left

⟨Landmarks⟩:

another truck

⟨Next Subgoal⟩:

take the next left at another truck

#### Output Information

When I have finished the turn left, I might see another truck 20 meters to the LEFT.

### Subgoal Judger

#### Input Information

⟨Current Subgoal⟩:

turn left

⟨Subgoal Memory⟩:

At step-1: I see a large tire and part of a truck's body; I turn left 15 degrees.

⟨Current Observation⟩:

Overall: I see a close-up view of a truck's rear section, including its tire and part of its body.

### Subgoal Judger

In Front: The back tire of the truck is directly in front of me, approximately 1 meter away. On my Left: The side of the truck is visible, extending to the left, with the truck's body occupying most of the left side of the image. On my Right: The truck's rear section, including part of its tailgate and body, is visible, occupying the right side of the image.

⟨Reference Imagination⟩:

When I have finished the turn left, I might see another truck 20 meters to the LEFT.

⟨Next Subgoal⟩:

take the next left at another truck

#### Output Information

subgoal: turn left

achieved: False

reason: The ⟨Current Observation⟩ indicates that the drone is still facing the rear section of the truck, with the truck's body occupying both the left and right sides of the image. The ⟨Subgoal Memory⟩ shows only a 15-degree turn, which is not sufficient to achieve a large turn ( $> 45$  degrees). The ⟨Reference Imagination⟩ suggests that another truck should be visible 20 meters to the left, which is not observed in the current state.

### Decision-Making

#### Input Information

⟨Instruction Memory⟩:

For ⟨turn left⟩:

At step-1: I see a large tire and part of a truck's body; I turn left 15 degrees.

⟨Current Observation⟩:

Overall: I see a close-up view of a truck's rear section, including its tire and part of its body.

In Front: The back tire of the truck is directly in front of me, approximately 1 meter away. On my Left: The side of the truck is visible, extending to the left, with the truck's body occupying most of the left side of the image. On my Right: The truck's rear section, including part of its tailgate and body, is visible, occupying the right side of the image.

⟨Collision Warning⟩:

\*\*MOVE\_FORWARD will collide with objects. ASCEND will collide with objects. DESCEND will collide with objects. \*\*

⟨Current Instruction⟩:

turn left and take the next left at another truck.

### Decision-Making

⟨Current Subgoal⟩:

turn left

⟨Valid Actions⟩:

1: MOVE\_FORWARD (5 meters)

2: TURN\_LEFT (15 degrees)

3: TURN\_RIGHT (15 degrees)

4: ASCEND (2 meters)

5: DESCEND (2 meters)

6: MOVE\_LEFT (5 meters)

7: MOVE\_RIGHT (5 meters)

#### Output Information

thought: Given the current subgoal to turn left and the observation that the truck's rear section is directly in front and to the sides, the most reasonable action is to turn left. Moving forward, ascending, or descending will result in a collision. Moving left or right will not align with the subgoal of turning left.

probabilities: {"1": 0.05, "2": 0.85, "3": 0.02, "4": 0.01, "5": 0.01, "6": 0.02, "7": 0.04}

selected\_action: 2

### Multi-Level Memory(Step Memory)

#### Input Information

⟨Current Observation⟩:

Overall: I see a close-up view of a truck's rear section, including its tire and part of its body. In Front: The back tire of the truck is directly in front of me, approximately 1 meter away. On my Left: The side of the truck is visible, extending to the left, with the truck's body occupying most of the left side of the image. On my Right: The truck's rear section, including part of its tailgate and body, is visible, occupying the right side of the image.

⟨Current Action⟩:

TURN\_LEFT (15 degrees)

⟨Landmarks⟩:

another truck | the back tire of the truck

#### Output Information

I see the back tire of the truck 1 meter in front; I turn left 15 degrees.

### Multi-Level Memory(Subgoal Memory)

At step-1: I see a large tire and part of a truck's body; I turn left 15 degrees.

At step-2: I see the back tire of the truck 1 meter in front; I turn left 15 degrees.

## References

- [1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 16
- [2] Yunpeng Gao, Zhigang Wang, Linglin Jing, Dong Wang, Xuelong Li, and Bin Zhao. Aerial vision-and-language navigation via semantic-topo-metric representation guided llm reasoning. *ArXiv*, abs/2410.08500, 2024. 9
- [3] Yunpeng Gao, Chenhui Li, Zhongrui You, Junli Liu, Zhen Li, Pengan Chen, Qizhi Chen, Zhonghan Tang, Liansheng Wang, Penghui Yang, Yiwen Tang, Yuhang Tang, Shuai Liang, Songyi Zhu, Ziqin Xiong, Yifei Su, Xinyi Ye, Jianan Li, Yan Ding, and Xuelong Li. Openfly: A versatile toolchain and large-scale benchmark for aerial vision-language navigation. *arXiv e-prints*, pages arXiv–2502, 2025. 10
- [4] Ross B. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 16
- [5] 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*, pages 4697–4708. PMLR, 2025. 10
- [6] Junnan Li, Dongxu Li, Silvio Savarese, and Steven C. H. Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International Conference on Machine Learning*, 2023. 16
- [7] Yuxing Long, Xiaoqi Li, Wenzhe Cai, and Hao Dong. Discuss before moving: Visual language navigation via multi-expert discussions, 2023. 18
- [8] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiayi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. 14, 24
- [9] Youcai Zhang, Xinyu Huang, Jinyu Ma, Zhaoyang Li, Zhaochuan Luo, Yanchun Xie, Yuzhuo Qin, Tong Luo, Yaqian Li, Siyi Liu, Yandong Guo, and Lei Zhang. Recognize anything: A strong image tagging model. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1724–1732, 2023. 18
- [10] Gengze Zhou, Yicong Hong, Zun Wang, Xin Eric Wang, and Qi Wu. Navgpt-2: Unleashing navigational reasoning capability for large vision-language models. In *European Conference on Computer Vision*, pages 260–278. Springer, 2024. 16