THUNDERBIRD: ADAPTIVE FAST-SLOW UAV NAVIGATION WITH SCENE COMPLEXITY AWARENESS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent UAV-VLN methods often employ complex VLM reasoning pipelines to improve navigation reliability, but apply them uniformly at every decision step—even in simple or unambiguous states where lightweight reasoning would suffice—resulting in unnecessary latency and computation. To address this, we propose a fast–slow navigation framework that separates high-level waypoint decision-making from low-level flight execution. At every navigation decision step, the slow system generates an intermediate waypoint and dynamically adjusts its VLM reasoning budget according to the difficulty of the current navigation state. The fast system then performs waypoint-to-waypoint motion planning, converting the selected waypoint into a safe and executable trajectory. To coordinate the fast and slow systems, we introduce the OVL scene complexity score that estimates the difficulty of the current navigation decision. Experimental results show that, compared with a conventional slow-system baseline that performs full reasoning at every navigation decision step, our method reduces the average VLM reasoning time by 49.4% and the flight time by 65.2%. Furthermore, compared with representative state-of-the-art methods, our approach achieves 2.57× the flight speed without compromising the navigation success rate.

1 INTRODUCTION

In Vision-and-Language Navigation (VLN), an embodied agent navigates to a target by interpreting natural-language instructions and visual observations (Anderson et al., 2018; Krantz et al., 2020). In recent years, VLN research has expanded from indoor navigation on predefined connectivity graphs to continuous environments that require realistic motion planning and low-level control (Krantz et al., 2020; Liu et al., 2023; Wang et al., 2025). We refer to VLN for unmanned aerial vehicles (UAVs) as UAV-VLN. In each task, the UAV receives a single natural-language instruction at the outset and uses its onboard visual observations to reach the described target; the instruction remains fixed throughout navigation. Because UAVs operate over large three-dimensional spaces, UAV-VLN tasks are typically long-range and involve target identification and search, three-dimensional route selection, local obstacle avoidance, and continuous flight control (Liu et al., 2023; Fan et al., 2023; Wang et al., 2025). To identify targets and select routes, most UAV-VLN methods use vision-language models (VLMs) to jointly interpret multi-view observations and language instructions and generate an intermediate waypoint for the next flight segment (Liu et al., 2023; Wang et al., 2025; Gao et al., 2026; Zhang et al., 2025b; Xu et al., 2026; Ning et al., 2026; Cheng et al., 2026). Some methods use parallel exploration, serial refinement, and multi-round reasoning to improve the reliability of navigation decisions (Cheng et al., 2026). However, these complex reasoning pipelines can substantially increase decision-making time.

Furthermore, existing methods usually use a fixed reasoning configuration throughout navigation (Wang et al., 2025; Cheng et al., 2026). They do not fully consider how the difficulty of the current navigation state changes during flight. During a long-range task, UAVs may pass through open areas, regions near buildings, narrow passages, and target-dense areas, where the environmental structure and direction-selection difficulty may differ considerably across navigation decision steps. When the scene structure is clear and the forward direction is easy to determine, one VLM call is often sufficient to generate a reliable intermediate waypoint, while additional candidate generation may introduce unnecessary computational overhead. In contrast, when occlusion is severe, the space is crowded, or the direction is difficult to determine, a fixed reasoning budget may be insufficient

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

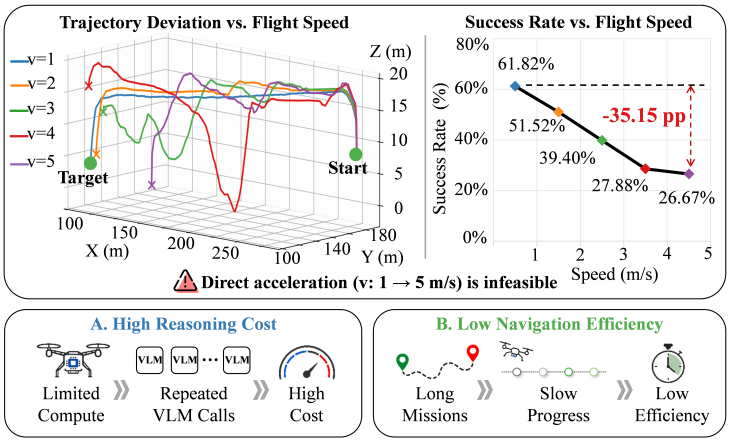


Figure 1: Motivation of Thunderbird. Simply increasing the flight speed in TravelUAV leads to greater trajectory fluctuations and lower navigation performance. A fixed number of VLM calls introduces redundant computation, while conservative flight strategies limit task execution efficiency.

for reliable decision-making. However, how to dynamically allocate the VLM reasoning budget according to the actual difficulty of the current navigation state remains underexplored. Therefore, simple navigation states should avoid redundant reasoning, while difficult states should receive more decision support. This balance is important for improving UAV-VLN efficiency without weakening the navigation decision-making capability of the VLM.

Efficient long-range navigation also depends on how quickly the UAV executes each generated waypoint. Simply raising the speed limit is not sufficient: in TravelUAV, it produces larger trajectory deviations and reduces navigation success, as shown in Fig. 1. This observation motivates a low-level motion planner that improves waypoint-execution efficiency while maintaining flight reliability.

To address these issues, we propose Thunderbird, a scene-complexity-aware fast–slow framework for UAV-VLN. At each navigation decision step, the slow system generates an intermediate waypoint using a VLM reasoning budget adapted to the difficulty of the current state, while the fast system plans and executes a continuous trajectory to that waypoint. We introduce OVL, named after its three components: Object Overlap, View Count Variation, and Large Object Count. This scene-complexity score jointly regulates the reasoning budget and flight speed, allowing Thunderbird to reduce decision-making time and improve flight efficiency without sacrificing navigation reliability.

In summary, our main contributions are as follows:

- We propose Thunderbird, a fast–slow navigation framework that connects high-level reasoning with low-level motion planning via intermediate waypoints.
- We introduce the OVL scene complexity score, which dynamically adjusts the VLM reasoning budget and flight speed according to the current navigation state.
- We conduct closed-loop evaluations on TravelUAV, demonstrating competitive navigation performance, increased flight speed, and reduced VLM inference time.

2 Related Work

2.1 UAV Motion Planning

UAV motion planning seeks safe, dynamically feasible trajectories under strict real-time constraints. Sampling-based and polynomial methods construct collision-free, smooth paths, represented by RRT* (Karaman & Frazzoli, 2011) and minimum-snap (Mellinger & Kumar, 2011). Online optimizers such as EGO-Planner (Zhou et al., 2021) enable rapid replanning in unknown environments, while chance-constrained MPC (Venkateswara Rao et al., 2026) handles uncertainty from dynamic obstacles. Recent learning-based approaches further use reinforcement learning (Ryou et al., 2026), perception-aware objectives (Qin et al., 2026), and world models (Zhang et al., 2026) to improve

time optimality and adaptability. Since these methods generally adopt fixed planning strategies, Thunderbird uses scene complexity to regulate both VLM reasoning effort and flight speed.

2.2 UAV Vision-Language Navigation

UAV-VLN extends motion planning by grounding language instructions in onboard visual observations for task-oriented 3D navigation. Data-centric works, including AerialVLN (Liu et al., 2023), OpenFly (Gao et al., 2026), and AirNav (Cai et al., 2026), improve the scale and realism of training and evaluation. Reasoning-oriented methods such as CityNavAgent (Zhang et al., 2025b), GeoNav (Xu et al., 2026), and LookasideVLN (Ning et al., 2026) support long-horizon navigation through hierarchical planning, structured spatial memory, and directional modeling. SkyVLN (Li et al., 2025) and ImagineUAV (Liu et al., 2026) further connect semantic decisions with executable trajectories through MPC or world-action modeling. However, most systems use fixed model-invocation and control pipelines. Thunderbird instead adapts high-level reasoning and low-level planning according to scene complexity.

2.3 Scene Complexity Measurement and Estimation

Scene complexity measurement converts structural, dynamic, and visual factors into comparable difficulty estimates. Navigation-oriented methods evaluate obstacle layout and traversability: BARN (Perille et al., 2020) and DynaBARN (Nair et al., 2022) define static and dynamic difficulty benchmarks, while Stratton et al. (2025) relate density, passage width, and crowd motion to navigation performance. Risk-oriented approaches model interactions explicitly; Yu et al. (2021) combine RSS with information entropy, and SOTIF Entropy (Peng et al., 2024) jointly captures collision risk and perception uncertainty. Image-based methods estimate visual complexity directly, with ICNet (Feng et al., 2023), DSP (Beche & Nedevschi, 2025), and Guo et al. (2026) targeting perceptual scoring, aerial scenes, and localized complexity. As existing metrics are rarely linked to planner selection, Thunderbird uses scene complexity to schedule its fast–slow navigation framework.

3 Method

3.1 Fast–Slow Navigation Framework

We propose Thunderbird, a fast–slow collaborative framework for UAV-VLN with scene complexity awareness. As shown in Fig. 2, Thunderbird divides the navigation task into high-level navigation decision-making and low-level flight execution. The slow system performs high-level navigation decision-making, while the fast system handles low-level flight execution. The slow system interprets instructions and multi-view images.

3.1.1 Slow System: High-Level Navigation Reasoning

As the high-level navigation decision module of Thunderbird, the slow system generates an intermediate waypoint based on the task instruction and the current environment. We use the publicly released pretrained model from TravelUAV (Wang et al., 2025) as the base navigation model. The model adopts Vicuna-7B (Zheng et al., 2023) as the language backbone and uses EVA-CLIP (Sun et al., 2023) to extract multi-view image features. A Q-Former (Li et al., 2023) then maps the visual features into the language representation space. The visual features and task instruction are jointly fed into the multimodal language model to represent task semantics, environmental information, and their spatial relationships.

Each task provides a single instruction Q, which remains fixed throughout navigation. At navigation decision step t, the slow system combines Q with the current multi-view images I_t. Existing work employs test-time scaling (TTS) to improve navigation decision quality through multi-round reasoning (Cheng et al., 2026), but typically uses a fixed TTS budget at every decision step. Thunderbird instead first evaluates the complexity C_t of the local scene from I_t. It then determines the number of VLM calls n_t, and hence the TTS budget, as a function of C_t:

$$n_t = h(C_t)$$

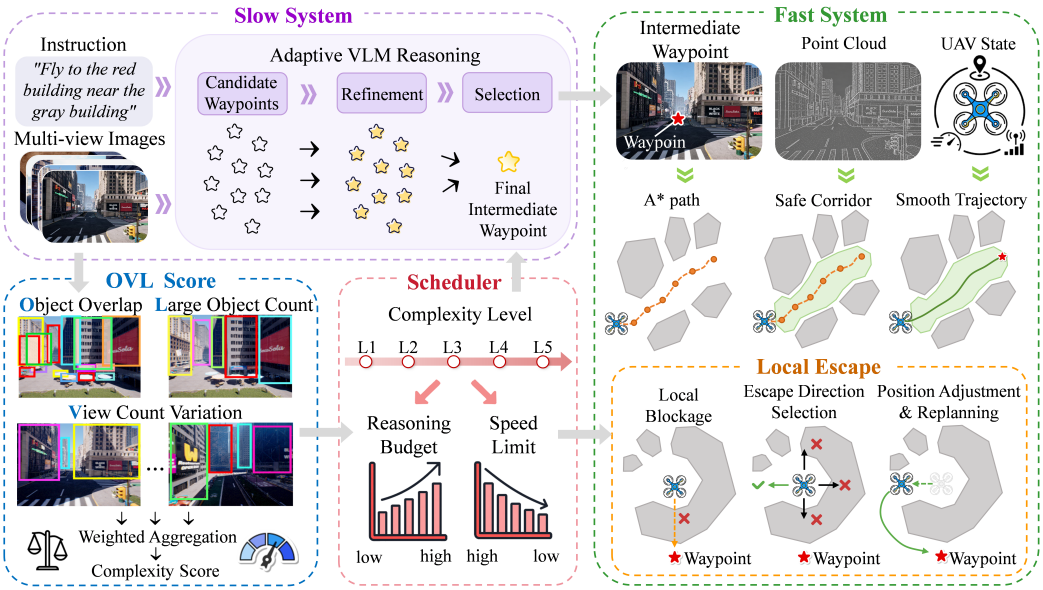


Figure 2: Overall framework of Thunderbird. The slow system generates an intermediate waypoint from the instruction and multi-view images through an adaptive VLM reasoning budget, while the fast system converts it into a safe trajectory using the point cloud and UAV state. The Local Escape module handles planning failures. OVL estimates scene complexity from Object Overlap, View Count Variation, and Large Object Count, and accordingly adjusts the reasoning budget and speed limit.

Under this budget, the slow system generates candidate intermediate waypoints, refines them, and selects the final result. The exact definition of C_t and the associated TTS-budget allocation strategy are detailed in a later subsection. The resulting intermediate waypoint at the current decision step is formulated as

$$w_t = S\left(Q, I_t, n_t\right)$$

where $S(\cdot)$ denotes the test-time scaling process of the slow system, and w_t is the final intermediate waypoint. This waypoint is then passed to the fast system as the target for local motion planning and flight control.

For scenes with clear environmental structures and unambiguous navigation directions, Thunderbird uses fewer VLM calls to reduce unnecessary inference overhead.

3.1.2 Fast System: Low-Level Motion Planning

As the low-level motion planning module of Thunderbird, the fast system generates safe and dynamically feasible flight trajectories (Ren et al., 2025). It uses the intermediate waypoint from the slow system and real-time local observations. The planner receives the real-time point cloud and the UAV motion state. It then updates the local occupancy representation and generates a collision-free trajectory from the current position to the intermediate waypoint.

During the navigation decision step t of the slow system, the fast system performs multiple local planning cycles for the intermediate waypoint w_t. Let $P_{t,j}$ denote the point cloud observation at planning cycle j, and let $x_{t,j}$ denote the UAV motion state. The continuous flight trajectory generated by the fast system is expressed as

$$T_{t,j} = F\left(w_t, P_{t,j}, x_{t,j}\right)$$

where $F(\cdot)$ denotes the local environment modeling and motion planning process. $T_{t,j}$ denotes the continuous flight trajectory generated at planning cycle j during navigation decision step t.

The fast system updates the local occupancy map from the point cloud and uses the current UAV position as the start and the intermediate waypoint from the slow system as the goal. It then searches for a collision-free A* path, constructs safe flight corridors along the path, and generates a smooth trajectory subject to the UAV velocity and acceleration constraints (Ren et al., 2025). The resulting position, velocity, and acceleration commands are sent to the low-level controller. Meanwhile, the Scheduler adjusts the flight speed limit according to the OVL complexity level.

When local obstacles cause consecutive planning failures, the Local Escape module temporarily adjusts the UAV position through a safe backward, lateral, or upward movement while retaining the original intermediate waypoint. The fast system then resumes local planning. If neither local replanning nor safe escape produces a feasible trajectory before the time limit, the task terminates due to a timeout. Further implementation details of occupancy mapping, corridor construction, trajectory generation, and local escape are provided in the appendix.

3.2 Scene Complexity Measure

3.2.1 OVL Complexity Metric

During flight, the UAV can obtain visual information from multiple directions at the same time. Compared with a single view, multi-view images provide more information about the environment. They reflect both the object density within each view and the environmental differences across directions. As suggested by its name, OVL comprises three measures: Object Overlap, View Count Variation, and Large Object Count, denoted by O_t, V_t, and L_t, respectively.

At navigation decision step t, the system first uses Grounding DINO (Liu et al., 2024) to detect objects in the RGB images from five views. Duplicate detections of the same object may lead to an overestimated complexity score. Therefore, the system removes detection boxes with highly similar class descriptions, locations, and scales. It retains genuine spatial overlap between different objects. After filtering, let $m_{t,k}$ be the number of objects in view k, where $k \in \{1, \ldots, 5\}$.

O_t measures the crowding of the current visual scene. For detection box $b_{t,k,r}$ in view k, we first calculate the ratio between its overlap with other detection boxes in the same view and its own area:

$$o_{t,k,r} = \min\left(\frac{\sum_{u \neq r} |b_{t,k,r} \cap b_{t,k,u}|}{|b_{t,k,r}|}, 1\right)$$

Here, $b_{t,k,r}$ is detection box r in view k at navigation decision step t. u refers to another detection box in the same view. $|b_{t,k,r}|$ is the area of the detection box. $|b_{t,k,r} \cap b_{t,k,u}|$ is the overlap area between the two detection boxes. A larger $o_{t,k,r}$ indicates more overlap between this object and other objects in the image. We then average the values over all detected objects in the five views:

$$O_t = \frac{\sum_{k=1}^{5} \sum_{r=1}^{m_{t,k}} o_{t,k,r}}{\sum_{k=1}^{5} m_{t,k}}$$

If no object is detected in any of the five views, we set $O_t = 0$.

V_t describes the environmental differences across viewing directions. We calculate the average number of detected objects over the five views and the corresponding standard deviation as

$$\bar{m}_t = \frac{1}{5}\sum_{k=1}^{5} m_{t,k}, \qquad V_t = \sqrt{\frac{1}{5}\sum_{k=1}^{5}(m_{t,k} - \bar{m}_t)^2}$$

Here, $m_{t,k}$ is the number of objects detected in view k at navigation decision step t. $\bar{m}_t$ is the average object count over the five views. V_t measures the variation in object counts across views. When the five views contain similar numbers of objects, V_t is small. This indicates that the objects are distributed relatively evenly across different directions. When the object counts differ greatly across views, V_t increases. This indicates clear differences in environmental complexity across flight directions.

L_t describes objects that occupy a large area in the field of view. For each detection box, the system calculates the ratio of its area to the full image area. A detection is classified as a large object when this ratio is at least 3%. Let $a_{t,k}$ be the number of large objects in view k. Then,

$$L_t = \max_{k \in \{1,\dots,5\}} a_{t,k}$$

We use the maximum rather than the average to preserve information from locally complex directions. Even when the other views are relatively open, L_t increases if one view contains many objects with large visual occupancy. This increases the estimated complexity of the current scene.

The three components have different value ranges. We first collect the distributions of O_t, V_t, and L_t over a large set of multi-view images. We then determine a normalization reference value for each component based on its typical range. Each component is divided by its reference value, and the result is clipped to $[0, 1]$:

$$\hat{O}_t = \min\left(\frac{O_t}{0.7}, 1\right), \quad \hat{V}_t = \min\left(\frac{V_t}{10.0}, 1\right), \quad \hat{L}_t = \min\left(\frac{L_t}{6.0}, 1\right)$$

Here, 0.7, 10.0, and 6.0 are the normalization reference values for O_t, V_t, and L_t. These values are obtained from statistics over the multi-view image data. The final OVL scene complexity score is defined as

$$C_t = 0.60\hat{O}_t + 0.30\hat{V}_t + 0.10\hat{L}_t$$

O_t has the highest weight. It mainly reflects crowding and occlusion in the visual scene. V_t describes environmental differences across flight directions. L_t reflects significant spatial occupancy in the local field of view. The final score satisfies $C_t \in [0, 1]$. A larger value indicates a more crowded visual scene and greater differences across directions. It also indicates greater difficulty for high-level navigation decision-making and local flight control.

3.2.2 Complexity-Guided Reasoning and Speed Scheduling

Thunderbird uniformly partitions the normalized complexity score into five levels, from L1 to L5, and jointly adjusts the VLM reasoning budget and flight speed. As scene complexity increases, the system uses different combinations of parallel candidate generation and serial refinement to increase the overall reasoning budget. Consequently, the number of VLM calls increases from 1 to 6, while the flight speed limit decreases from 3.0 m/s to 1.0 m/s. Detailed reasoning configurations for each complexity level are provided in the appendix.

A smaller reasoning budget and a higher speed limit reduce inference and execution overhead in simple scenes. In complex scenes, a larger reasoning budget provides stronger decision support, while a lower speed limit allows more time for local replanning and control correction. This mechanism enables an adaptive balance among navigation efficiency, decision reliability, and flight safety.

4 Experiments

4.1 Experimental Setup

We conduct closed-loop UAV-VLN experiments on the AirSim (Shah et al., 2018) simulation platform using the TravelUAV dataset (Wang et al., 2025). The evaluation includes the Test Seen (TS), Unseen Object (UO), and Unseen Map (UM) test sets (Wang et al., 2025). Thunderbird directly uses the publicly released pretrained model (Wang et al., 2025) without additional training or parameter fine-tuning. For each navigation task, the UAV starts from a given initial position. It continuously generates intermediate waypoints from the instruction and multi-view images. The UAV then executes the flight until it reaches the target or meets a task termination condition. OVL parameters are selected on a development subset of the TravelUAV training split and fixed before evaluation on TS, UO, and UM.

We evaluate the system in terms of navigation performance and runtime efficiency. Navigation performance is measured using Navigation Error (NE), Success Rate (SR), Oracle Success Rate (OSR), and Success weighted by Path Length (SPL). Runtime efficiency is measured using Flight

Table 1: Navigation performance comparison on the TS, UO, and UM test sets.

Method	TS				UO				UM			
	NE	SR	OSR	SPL	NE	SR	OSR	SPL	NE	SR	OSR	SPL
Random	222.20	0.14	0.21	0.07	260.14	0.16	0.16	0.16	202.98	0.00	0.00	0.00
Fixed Action	188.61	2.27	8.16	1.40	212.84	3.66	9.54	2.16	180.47	0.52	2.61	0.39
CMA	135.73	8.37	18.72	7.90	155.79	9.06	16.06	8.68	141.68	2.30	10.02	2.16
TravelUAV	110.99	22.94	44.92	19.50	124.79	24.48	41.49	21.53	143.08	5.32	15.45	4.61
TravelUAV_TTS	106.32	24.96	47.39	20.93	121.11	25.76	43.08	22.66	135.58	6.26	16.81	5.26
NavFoM	93.05	29.17	49.24	25.03	108.04	29.83	47.99	27.20	125.10	6.30	18.95	5.68
NeuroKalman	71.56	25.86	58.73	22.43	71.01	32.48	60.82	28.50	**100.32**	8.34	**34.15**	7.12
LongFly	60.02	36.39	**65.87**	31.07	**66.74**	**43.87**	64.56	**38.39**	108.32	11.27	30.27	9.32
Thunderbird w/o OVL	64.61	35.97	59.59	32.66	72.69	37.68	61.21	35.59	121.41	10.02	23.59	8.92
Thunderbird	**58.28**	**37.73**	62.55	**34.27**	70.12	40.38	**65.18**	38.15	115.67	**11.48**	26.10	**10.23**

Speed, Flight Time, VLM Time, and Task Time. Flight Speed is measured in meters per second ($\mathrm{m/s}$), while Flight Time, VLM Time, and Task Time are measured in seconds (s).

4.2 Main Results

We compare Thunderbird with conventional baselines, representative UAV-VLN methods, and recent approaches, including TravelUAV_TTS (Cheng et al., 2026), NavFoM (Zhang et al., 2025a), NeuroKalman (Tang et al., 2026), and LongFly (Jiang et al., 2025), with the results reported in Table 1. We use TravelUAV_TTS as the baseline because it adopts the same pretrained navigation model while increasing VLM calls through test-time scaling. Thunderbird consistently outperforms this baseline on the TS, UO, and UM test sets, supporting the effectiveness of the proposed fast–slow framework.

Thunderbird also remains competitive with stronger recent methods. Among the compared methods, it achieves the best NE, SR, and SPL on TS, the highest OSR on UO with an SPL close to the best result, and the highest SR and SPL on UM. More importantly, it maintains strong navigation performance while improving flight speed and reducing inference overhead.

Comparing Thunderbird w/o OVL with the complete model shows that OVL further improves NE, SR, OSR, and SPL across all three test sets. This indicates that jointly adapting the VLM reasoning budget and flight speed to scene complexity improves navigation stability and reliability beyond the fast–slow framework alone.

4.3 Efficiency Analysis

While maintaining competitive navigation performance, we further evaluate the runtime efficiency of Thunderbird in flight execution and model inference. All methods are evaluated on all scenes in the TS test set. However, the efficiency metrics are computed only over the set of tasks successfully completed by all methods. This reduces interference from repeated wandering, ineffective flight, or timeout termination in failed tasks. It therefore ensures a fair comparison across different methods. The results are shown in Table 2.

Compared with TravelUAV_TTS (Cheng et al., 2026), Thunderbird increases the average flight speed from 0.79 m/s to 2.03 m/s. It also reduces Flight Time, VLM Time, and Task Time by 65.2%, 49.4%, and 56.9%, respectively. These improvements mainly come from continuous trajectory execution, on-demand allocation of the VLM reasoning budget, and a more efficient observation and decision-making process. Together, these designs reduce the redundant overhead caused by fixed test-time scaling and frequent system interactions.

Thunderbird w/o OVL achieves higher baseline runtime efficiency. After adding OVL, the system increases the VLM reasoning budget and appropriately lowers the flight speed in complex scenes. Compared with Thunderbird w/o OVL, the complete model uses more VLM Time and slightly lowers the flight speed, while improving all navigation metrics across TS, UO, and UM. This result is consistent with the intended behavior of OVL: instead of maximizing raw execution speed, it selectively spends additional reasoning effort to improve navigation performance in complex states. Despite the added overhead, Thunderbird's Task Time remains substantially lower than those of

Table 2: Runtime efficiency comparison on the successful tasks of the TS test set.

Method	Speed (m/s)	Flight Time	VLM Time	Task Time
TravelUAV	0.78	353.24	47.78	1812.38
TravelUAV_TTS	0.79	339.30	290.51	2044.92
Thunderbird w/o OVL	**2.26**	**107.61**	**40.43**	**753.31**
Thunderbird	2.03	118.02	147.14	882.17

Table 3: Ablation of the OVL components on the TS test set.

Variant	NE	SR	OSR	SPL
Thunderbird w/o OVL	64.61	35.97	59.59	32.66
w/o O	63.52	36.18	60.16	32.98
w/o V	60.45	36.53	61.21	33.14
w/o L	60.76	37.30	60.72	34.06
Thunderbird	**58.28**	**37.73**	**62.55**	**34.27**

TravelUAV (Wang et al., 2025) and TravelUAV_TTS, indicating a favorable balance between navigation performance and runtime efficiency.

4.4 Ablation of OVL Components

To analyze the contribution of each scene complexity component in OVL, we separately remove O_t, View Count Variation V_t, and Large Object Count L_t on the TS test set. For each ablation variant, the weights of the remaining components are proportionally renormalized to sum to one. We compare these with Thunderbird w/o OVL and the complete Thunderbird method. The results are shown in Table 3.

Compared with Thunderbird w/o OVL, introducing any scene complexity component improves navigation performance. The complete OVL achieves the best NE, SR, OSR, and SPL. Removing O_t causes the largest overall performance drop. This indicates that O_t is an important factor for estimating scene complexity. Removing V_t or L_t also leads to different degrees of performance degradation. This shows that variation across views and the presence of large objects provide complementary scene complexity information. These results verify the effectiveness of all three components and the necessity of using them together.

4.5 Weight Analysis of OVL

To determine the weights of the three OVL components, we construct a development subset using multi-view observations sampled exclusively from the TravelUAV training split. This subset does not overlap with the TS, UO, or UM evaluation sets. We evaluate different weight combinations on the development subset and use AUC as the selection criterion. Fig. 3 shows the AUC results under different weight configurations.

When the weights of O_t, V_t, and L_t are set to 0.6, 0.3, and 0.1, respectively, OVL achieves the highest AUC of 0.73. This result indicates that O_t is the main factor for estimating scene complexity. V_t provides important complementary information, while L_t further describes local visual occupancy.

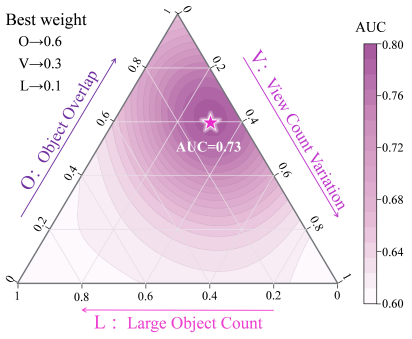
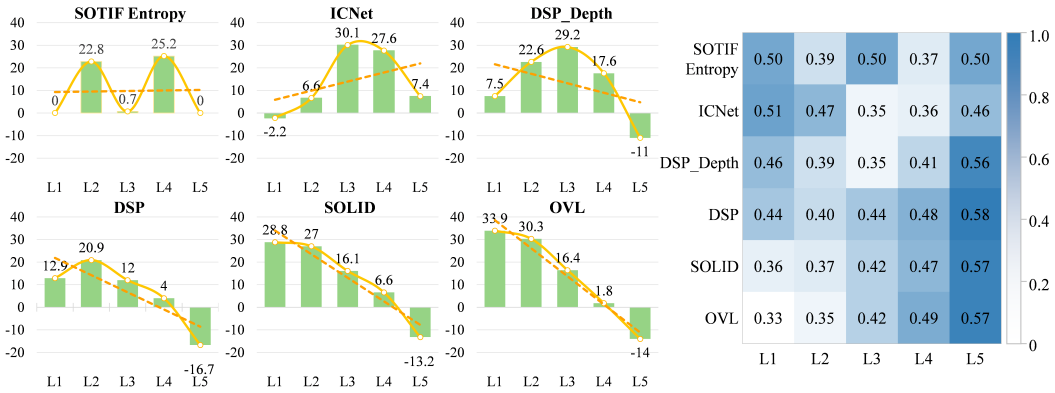


Figure 3: AUC of different OVL weight combinations on the development subset sampled from the training split.

4.6 Scene Complexity Metric Comparison

4.6.1 Comparison of Success and Failure Distributions

To evaluate OVL, we compare it with SOTIF Entropy (Peng et al., 2024), ICNet (Feng et al., 2023), DSP (Beche & Nedevschi, 2025), SOLID (Law & Deng, 2024), and DSP_Depth, which extends DSP with depth-gradient information. We compute each metric on all TS scenes and divide the observation frames into five complexity levels according to their scores. For each complexity level, we calculate the failure rate as the proportion of frames associated with failed tasks among all frames assigned to that level. Fig. 4(a) shows the difference between the success rate and failure rate within each complexity level, while Fig. 4(b) shows the failure rate within each level. A reliable complexity metric should exhibit a decreasing trend in the former and an increasing trend in the latter from L1 to L5.



(a) Success–Failure Proportion Gap (b) Failed-Task Proportion

Figure 4: Comparison of scene complexity metrics on the TS test set. (a) Difference between the success rate and failure rate within each complexity level. (b) Failure rate within each complexity level from L1 to L5.

OVL shows the clearest monotonic trends. Other metrics show smaller changes or fluctuations across adjacent levels. These results indicate that OVL more effectively assigns observations associated with successful navigation to lower complexity levels and observations associated with a higher risk of navigation failure to higher complexity levels.

4.6.2 Correlation Analysis

To quantitatively analyze the relationship between scene complexity metrics and navigation success or failure, we associate the complexity score of each observation frame with the final result of its corresponding task across all scenes in the TS test set. AUC measures the ability of the complexity score to distinguish successful-task frames from failed-task frames. Pearson r (PCC) and Spearman ρ (SRCC) measure the linear correlation and rank correlation between the complexity score and the navigation outcome, respectively. Hedges' g measures the difference between the complexity-score distributions of the two types of observation frames.

Table 4: Correlation analysis between scene complexity metrics and navigation outcomes on the TS scenes.

Method	AUC	PCC	SRCC	Hedges' g
SOTIF Entropy	0.40	−0.09	−0.17	−0.18
ICNet	0.43	−0.11	−0.12	−0.22
DSP_Depth	0.46	−0.05	−0.07	−0.10
DSP	0.50	0.01	0.00	0.02
SOLID	0.68	0.33	0.32	0.70
OVL	**0.71**	**0.37**	**0.37**	**0.78**

As shown in Table 4, OVL achieves the best results on all four metrics. This indicates that its complexity scores are more consistent with actual navigation success and failure. Compared with the other metrics, OVL shows the strongest association with final task outcomes. It assigns higher complexity scores to observations that are more likely to lead to navigation failure. SOLID achieves the second-best results, while the other metrics show relatively weaker ability to distinguish scene complexity and weaker correlations with the actual navigation outcomes.

5 Conclusion

This paper proposes Thunderbird, a collaborative framework with fast and slow systems for UAV-VLN with scene complexity awareness. The slow system performs language-guided high-level navigation reasoning, while the fast system performs low-level motion planning to convert the intermediate waypoint into a continuous, safe, and executable flight trajectory. To handle scenes with different complexity levels during flight, we further design the OVL metric to dynamically adjust the VLM reasoning budget and UAV flight speed. Experimental results show that Thunderbird maintains competitive navigation performance across different test settings. It also substantially increases flight speed and reduces VLM inference overhead and time cost. Overall, allocating the VLM reasoning budget on demand and adjusting the flight strategy according to scene complexity provide a feasible solution for building efficient and practical UAV-VLN systems.

Reproducibility statement

We provide the experimental setup, evaluation protocol, OVL definitions, and key implementation details required to reproduce the main results. Additional details on complexity-aware scheduling, motion planning, and Local Escape are included in the appendix.

Ethics statement

Our experiments are conducted in simulation and do not involve human-subject studies.

AI use statement

We used generative AI tools only for language polishing and literature retrieval. All AI-assisted content and retrieved references were reviewed and verified by the authors, who take full responsibility for the final paper.

References

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3674–3683, 2018.

Radu Beche and Sergiu Nedevschi. ClaraVid: A holistic scene reconstruction benchmark from aerial perspective with delentropy-based complexity profiling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 26015–26025, October 2025.

Hengxing Cai, Yijie Rao, Ligang Huang, Zanyang Zhong, Jinhan Dong, Jingjun Tan, Changhao Nai, Jue Hou, Wenhao Lu, and Renxin Zhong. AirNav: A large-scale UAV vision-and-language navigation dataset with natural and diverse instructions, 2026. URL https://arxiv.org/abs/2601.03707.

Feinan Cheng, Dongliang Xu, Wenli Nong, Zhiheng Zhang, Ang Liu, Tianyu Wang, and Yue Yao. No training, better flights: Test-time scaled VLMs for UAV navigation, 2026.

Yue Fan, Winson Chen, Tongzhou Jiang, Chun Zhou, Yi Zhang, and Xin Wang. Aerial vision-and-dialog navigation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 3043–3061, Toronto, Canada, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.190.

Tinglei Feng, Yingjie Zhai, Jufeng Yang, Jie Liang, Deng-Ping Fan, Jing Zhang, Ling Shao, and Dacheng Tao. IC9600: A benchmark dataset for automatic image complexity assessment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(7):8577–8593, 2023. doi: 10.1109/TPAMI.2022.3232328.

Yunpeng Gao, Chenhui Li, Zhongrui You, Junli Liu, Zhen Li, Pengan Chen, Qizhi Chen, Zhonghan Tang, Liansheng Wang, Penghui Yang, Yiwen Tang, Yuhang Tang, Shuai Liang, Songyi Zhu, Ziqin Xiong, Yifei Su, Xinyi Ye, Jianan Li, Yan Ding, Dong Wang, Xuelong Li, Zhigang Wang, and Bin Zhao. OpenFly: A comprehensive platform for aerial vision-language navigation, 2026. URL https://arxiv.org/abs/2502.18041.

Xiaoying Guo, Liang Li, Tao Yan, Lu Wang, and Yuhua Qian. Aligning computational and human perceptions of image complexity: A dual-task framework for prediction and localization. *Journal of Visual Communication and Image Representation*, 115:104686, 2026. doi: 10.1016/j.jvcir.2025.104686. URL https://www.sciencedirect.com/science/article/pii/S1047320325003001.

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

Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011. doi: 10.1177/0278364911406761. URL https://doi.org/10.1177/0278364911406761.

Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *Computer Vision – ECCV 2020*, volume 12373 of *Lecture Notes in Computer Science*, pp. 104–120. Springer, 2020. doi: 10.1007/978-3-030-58604-1_7.

Hei Law and Jia Deng. Label-free synthetic pretraining of object detectors. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 946–956, 2024.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19730–19742. PMLR, 2023.

Tianshun Li, Tianyi Huai, Zhen Li, Yichun Gao, Haoang Li, and Xinhu Zheng. SkyVLN: Vision-and-language navigation and NMPC control for UAVs in urban environments. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 17199–17206, 2025. doi: 10.1109/IROS60139.2025.11247609.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. In *Computer Vision – ECCV 2024*, volume 15105 of *Lecture Notes in Computer Science*, pp. 38–55. Springer, 2024. doi: 10.1007/978-3-031-72970-6_3.

Shubo Liu, Hongsheng Zhang, Yuankai Qi, Peng Wang, Yanning Zhang, and Qi Wu. AerialVLN: Vision-and-language navigation for UAVs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15384–15394, October 2023. doi: 10.1109/ICCV51070.2023.01411.

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

Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, pp. 2520–2525, 2011. doi: 10.1109/ICRA.2011.5980409.

Anirudh Nair, Fulin Jiang, Kang Hou, Zifan Xu, Shuozhe Li, Xuesu Xiao, and Peter Stone. DynaBARN: Benchmarking metric ground navigation in dynamic environments. In *2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 347–352, 2022. doi: 10.1109/SSRR56537.2022.10018758.

Yuwei Ning, Ganlong Zhao, Yipeng Qin, Si Liu, Yang Liu, Liang Lin, and Guanbin Li. Lookaside-VLN: Direction-aware aerial vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 32441–32450, June 2026.

Liang Peng, Boqi Li, Wenhao Yu, Kai Yang, Wenbo Shao, and Hong Wang. SOTIF entropy: Online SOTIF risk quantification and mitigation for autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 25(2):1530–1546, 2024. doi: 10.1109/TITS.2023.3322166. URL https://doi.org/10.1109/TITS.2023.3322166.

Daniel Perille, Abigail Truong, Xuesu Xiao, and Peter Stone. Benchmarking metric ground navigation. In *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 116–121, 2020. doi: 10.1109/SSRR50563.2020.9292572.

Chao Qin, Jiaxu Xing, Rudolf Reiter, Angel Romero, Yifan Lin, Hugh H.-T. Liu, and Davide Scaramuzza. Perception-aware time-optimal planning for quadrotor waypoint flight, 2026. URL https://arxiv.org/abs/2603.04305.

Yunfan Ren, Fangcheng Zhu, Guozheng Lu, Yixi Cai, Longji Yin, Fanze Kong, Jiarong Lin, Nan Chen, and Fu Zhang. Safety-assured high-speed navigation for MAVs. *Science Robotics*, 10(98): eado6187, 2025. doi: 10.1126/scirobotics.ado6187.

Gilhyun Ryou, Geoffrey Wang, and Sertac Karaman. Multi-fidelity reinforcement learning for time-optimal quadrotor re-planning. *The International Journal of Robotics Research*, 45(4): 645–669, 2026. doi: 10.1177/02783649251364393. URL https://doi.org/10.1177/02783649251364393.

Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. AirSim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, pp. 621–635. Springer, 2018. doi: 10.1007/978-3-319-67361-5_40.

Andrew Stratton, Kris Hauser, and Christoforos Mavrogiannis. Characterizing the complexity of social robot navigation scenarios. *IEEE Robotics and Automation Letters*, 10(1):184–191, 2025. doi: 10.1109/LRA.2024.3502060.

Quan Sun, Yuxin Fang, Ledell Wu, Xinlong Wang, and Yue Cao. EVA-CLIP: Improved training techniques for CLIP at scale, 2023.

Yin Tang, Jiawei Ma, Jinrui Zhang, Alex Jinpeng Wang, and Deyu Zhang. Mitigating error accumulation in continuous navigation via memory-augmented kalman filtering, 2026.

D. M. K. K. Venkateswara Rao, H. Habibi, and H. Voos. Fast trajectory optimization with time-varying chance-constrained model predictive control of quadcopters for dynamic collision avoidance. *Aerospace Science and Technology*, 174:111815, 2026. doi: 10.1016/j.ast.2026.111815.

Xiangyu Wang, Donglin Yang, Ziqin Wang, Hohin Kwan, Jinyu Chen, Wenjun Wu, Hongsheng Li, Yue Liao, and Si Liu. Towards realistic UAV vision-language navigation: Platform, benchmark, and methodology. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=rUvCIvI4eB.

Haotian Xu, Yue Hu, Chen Gao, Zhengqiu Zhu, Yong Zhao, and Quanjun Yin. GeoNav: Empowering MLLMs with dual-scale geospatial reasoning for language-goal aerial navigation. *Pattern Recognition*, 177:113365, 2026. doi: 10.1016/j.patcog.2026.113365. URL https://www.sciencedirect.com/science/article/pii/S0031320326003304.

Rongjie Yu, Yin Zheng, and Xiaobo Qu. Dynamic driving environment complexity quantification method and its verification. *Transportation Research Part C: Emerging Technologies*, 127: 103051, 2021. doi: 10.1016/j.trc.2021.103051. URL https://www.sciencedirect.com/science/article/pii/S0968090X21000814.

Jiazhao Zhang, Anqi Li, Yunpeng Qi, Minghan Li, Jiahang Liu, Shaoan Wang, Haoran Liu, Gengze Zhou, Yuze Wu, Xingxing Li, Yuxin Fan, Wenjun Li, Zhibo Chen, Fei Gao, Qi Wu, Zhizheng Zhang, and He Wang. Embodied navigation foundation model, 2025a.

Weichen Zhang, Chen Gao, Shiquan Yu, Ruiying Peng, Baining Zhao, Qian Zhang, Jinqiang Cui, Xinlei Chen, and Yong Li. CityNavAgent: Aerial vision-and-language navigation with hierarchical semantic planning and global memory. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31292–31309, Vienna, Austria, July 2025b. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.1511. URL https://aclanthology.org/2025.acl-long.1511/.

Xinhong Zhang, Runqing Wang, Yunfan Ren, Ding Yu, Boyu Zhou, Jian Sun, Fang Deng, Jie Chen, and Gang Wang. MAD: Mapping-aware world models for agile quadrotor flight, 2026. URL https://arxiv.org/abs/2606.04534.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623, 2023.

Xin Zhou, Zhepei Wang, Hongkai Ye, Chao Xu, and Fei Gao. EGO-Planner: An ESDF-free gradient-based local planner for quadrotors. *IEEE Robotics and Automation Letters*, 6(2):478–485, 2021. doi: 10.1109/LRA.2020.3047728.

A Additional Qualitative Examples of Speed-Induced Navigation Instability

In the motivation section, we show that increasing the flight speed degrades closed-loop UAV navigation stability. Fig. 5 provides additional qualitative cases under different speed settings. These examples are selected from the same 165 navigation tasks used in the main experiments, where the flight speed is varied from 1 to 5 m/s. As the speed increases, tracking errors accumulate more rapidly, leading to larger trajectory deviations, unstable altitude changes, and navigation failures.

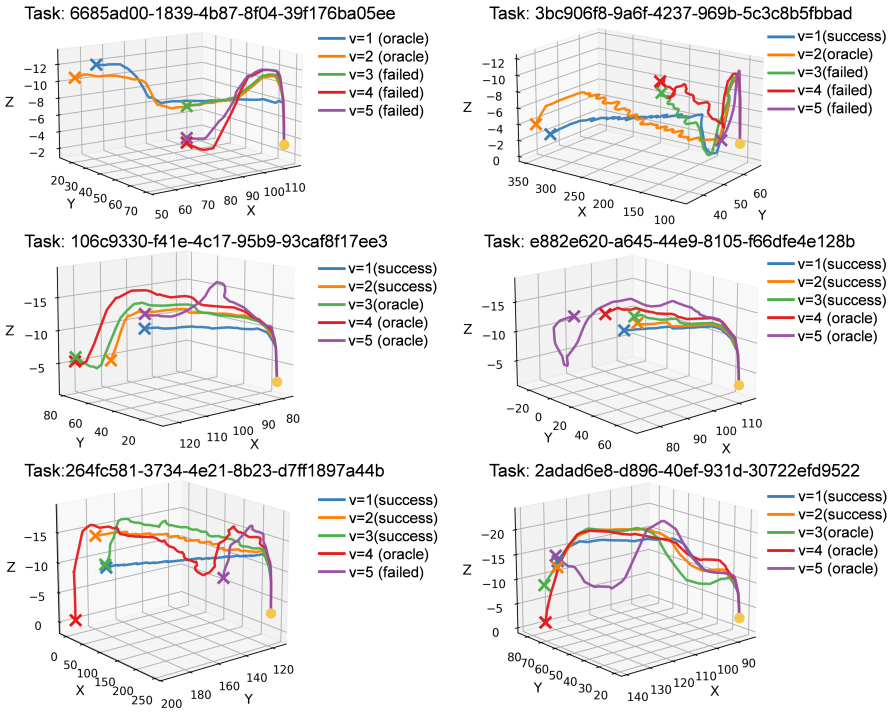


Figure 5: Representative UAV trajectories under different flight speed limits, showing increased trajectory deviation and navigation instability at higher speeds.

B Fast System: Low-Level Motion Planning

B.1 Local Motion Planning Pipeline

The fast system first builds a three-dimensional local occupancy grid map from the real-time point cloud. The map converts discrete three-dimensional measurements into a spatial representation for collision checking and path search. The UAV has a finite body size. If it is treated as a point during planning, a path may pass too close to an obstacle even without intersecting it. Therefore, the system inflates the obstacles in the map according to the UAV size and a predefined safety distance. This step reserves sufficient safety clearance for actual flight.

In the inflated local map, the system uses the current UAV position as the start and the stage subgoal waypoint from the slow system as the goal. It then uses the A* algorithm to search for a collision-free path. The A* path consists of discrete path points. It represents a feasible route to the target but cannot directly satisfy the requirements of smooth UAV flight. Therefore, the system uses CIRI to construct safe flight corridors along the A* path (Ren et al., 2025). The discrete path is first divided into several consecutive path segments. Each path segment is then expanded into the surrounding free space. The expansion continues until the segment approaches the boundaries of the inflated obstacles. This process forms a set of connected safe flight corridors that do not intersect obstacles.

Within the safe flight corridors, the system further generates a smooth and continuous flight trajectory. It also considers the velocity and acceleration constraints of the UAV. The resulting position,

velocity, and acceleration commands are then sent to the low-level controller. The flight speed limit is dynamically adjusted according to the current scene complexity. As the point-cloud observations and UAV motion state are continuously updated, the fast system repeats four operations: map updating, A* path search, safe flight corridor construction, and continuous trajectory generation. It can therefore adjust the flight trajectory in time and avoid newly observed obstacles.

B.2 Local Escape

To handle consecutive local planning failures, we introduce the Local Escape module. If the UAV is blocked by local obstacles at the current stage, the fast system may fail to produce a feasible plan for several consecutive planning cycles. This may also occur when no feasible passage is available near the current UAV position. In this case, the system retains the original stage subgoal waypoint. It then attempts to adjust the UAV position through short backward, lateral, or upward movements.

The Local Escape module first uses the local occupancy map to remove candidate directions with collision risks. It then evaluates the remaining directions using a combined score. The score considers three factors: the relative distance to the original target after the position adjustment, the deviation between the candidate direction and the target direction, and the number of previous planning failures along that direction. Let q_i be the UAV position after moving along candidate direction i. Its score is defined as follows:

$$J_i = w_d \frac{\|q_i - s_w\|}{\|p_{t,j} - s_w\|} + w_a \left(1 - \cos\theta_i\right) + w_f f_i$$

Here, $p_{t,j}$ is the current UAV position, s_w is the original stage subgoal waypoint generated by the slow system, and q_i is the position after moving along direction i. The distance ratio measures whether candidate position i is closer to the original stage subgoal waypoint than the current position. A smaller value means that the adjusted position is closer to the original stage subgoal waypoint. θ_i is the angle between candidate movement direction i and the direction toward the original stage subgoal waypoint. When the two directions are more closely aligned, $1 - \cos\theta_i$ is smaller. A larger directional deviation results in a higher cost. f_i is the number of previous planning failures along direction i. This term reduces the priority of directions that have repeatedly failed. w_d, w_a, and w_f are the weights for target distance, directional deviation, and the number of previous failures, respectively.

The system selects the direction with the lowest score from all candidate directions that satisfy the safety constraints:

$$i^* = \arg\min_i J_i$$

The UAV then moves along direction i^* to adjust its position.

C Complexity-Aware Reasoning and Speed Scheduling

Table 5: Complexity-guided adaptive adjustment strategy.

Complexity Range	Complexity Level	Reasoning Configuration	VLM Calls	Speed Limit
$[0.0, 0.2)$	L1	1×1	1	3.0
$[0.2, 0.4)$	L2	2×1	2	2.5
$[0.4, 0.6)$	L3	3×1	3	2.0
$[0.6, 0.8)$	L4	2×2	4	1.5
$[0.8, 1.0]$	L5	3×2	6	1.0

Thunderbird uniformly partitions the normalized complexity score into five levels and jointly adjusts the VLM reasoning budget and flight speed, as shown in Table 5. The reasoning configuration is represented as the number of parallel candidates multiplied by the serial refinement depth. The flight speed limit is measured in meters per second ($\mathrm{m/s}$).

In low-complexity scenes, the slow system uses a single candidate without serial refinement, reducing unnecessary inference overhead. As the complexity level increases from L1 to L5, Thunderbird uses different combinations of parallel candidate generation and serial refinement to progressively increase the overall VLM reasoning budget. Consequently, the number of VLM calls increases progressively from 1 to 6. Meanwhile, the fast system reduces the speed limit from 3.0 to 1.0 m/s as scene complexity increases. Higher speeds improve task execution efficiency in open scenes, whereas lower speeds provide additional time for local replanning and control correction in crowded or directionally ambiguous environments.

By jointly adapting reasoning effort and flight speed, Thunderbird avoids redundant computation in simple scenes while providing stronger decision support and more conservative execution in complex scenes. This enables an adaptive balance among navigation efficiency, decision reliability, and flight safety.