# *OpenFly*: A COMPREHENSIVE PLATFORM FOR AERIAL VISION-LANGUAGE NAVIGATION

**Anonymous authors**
Paper under double-blind review

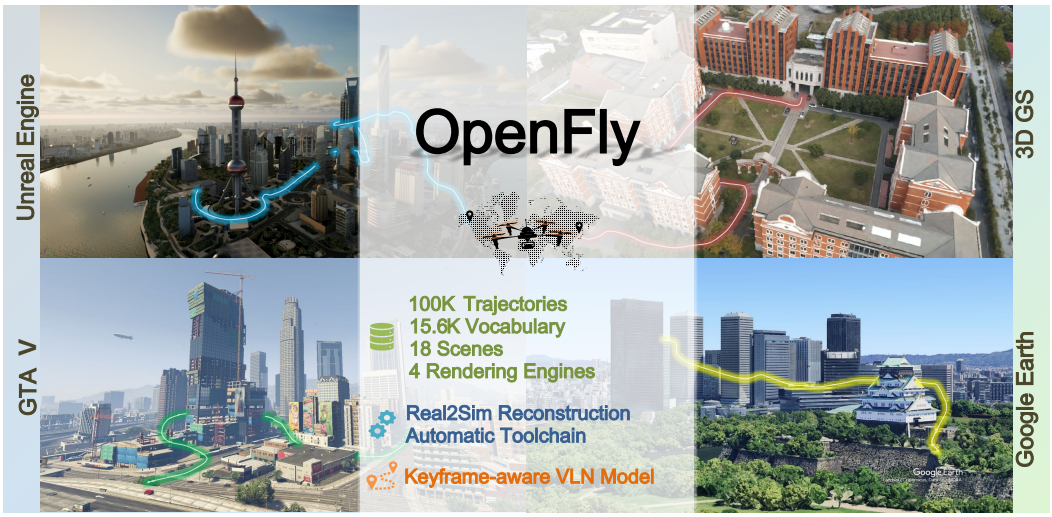



Figure 1: Overview of OpenFly. This work consists of (1) the integration of 4 rendering engines, significantly enhancing the diversity of scenario resources for aerial vision-language navigation; (2) an automatic data generation toolchain, eliminating reliance on labor-intensive annotations; (3) the largest aerial VLN dataset to date, comprising 100K trajectories; and (4) a keyframe-aware VLN model, achieving superior performance in both simulated and real-world scenes.

## ABSTRACT

Aerial Vision-Language Navigation (VLN) seeks to guide UAVs by leveraging language instructions and visual cues, establishing a new paradigm for human-UAV interaction. However, the collection of VLN data demands extensive human effort to construct trajectories and corresponding instructions, hindering the development of large-scale datasets and capable models. To address this problem, we propose **OpenFly**, a comprehensive platform for aerial VLN. Firstly, OpenFly integrates 4 rendering engines and advanced techniques for diverse environment simulation, including Unreal Engine, GTA V, Google Earth, and 3D Gaussian Splatting (3D GS). Particularly, 3D GS supports real-to-sim rendering, further enhancing the realism of our environments. Secondly, we develop a highly automated toolchain for aerial VLN data collection, streamlining point cloud acquisition, scene semantic segmentation, flight trajectory creation, and instruction generation. Thirdly, based on the toolchain, we construct a large-scale aerial VLN dataset with 100k trajectories, covering samples of diverse scenarios and assets across 18 scenes. Moreover, we propose OpenFly-Agent, a keyframe-aware VLN model emphasizing key observations to promote performance and reduce computations. For benchmarking, extensive experiments and analyses are conducted, where our navigation success rate outperforms others by 14.0% and 7.9% on the seen and unseen scenarios, respectively. The toolchain, dataset, and codes will be open-sourced.

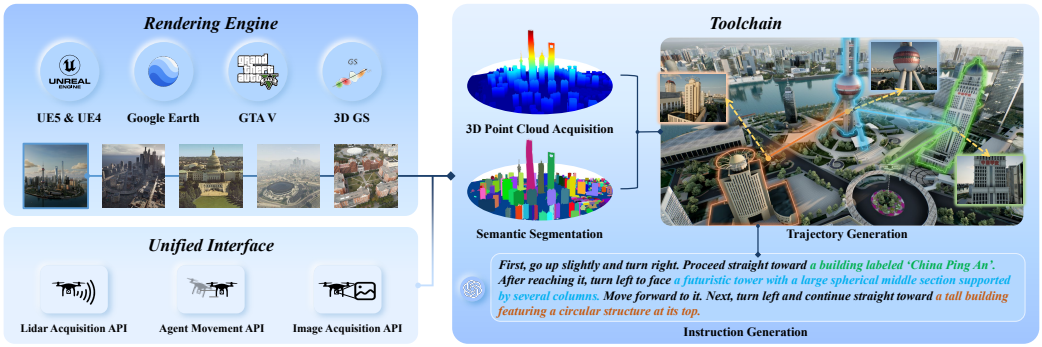



Figure 2: Framework of the automatic data generation. Multiple rendering engines are integrated to provide diverse, high-quality scenes. Built on these, several interfaces and tools are developed to enable automated generation of trajectories and instructions.

## 1 Introduction

Embodied AI has drawn growing research attention, where vision-language navigation (VLN) emerging as a core task that navigate agents to a target location according to linguistic instructions and visual observations. A number of benchmark datasets have been established, *e.g.,* Touch-Down (Chen et al., 2019), REVERIE (Qi et al., 2020), R2R (Anderson et al., 2018), RxR (Ku et al., 2020), CVDN (Thomason et al., 2019), VLN-CE (Krantz et al., 2020), and LANI (Misra et al., 2018), which have significantly advanced the development of VLN methods (Long et al., 2024; Hong et al., 2022; Wang et al., 2024c; Chen et al., 2022; Zhang et al., 2024; Cao et al., 2025; Guo et al., 2025; Ma et al., 2025). Nevertheless, existing efforts primarily target indoor or ground-based agents, while unmanned aerial vehicles (UAVs), crucial for aerial photography, rescue operations, and cargo transport, remain unexplored.

Most recently, AerialVLN (Liu et al., 2023) and OpenUAV (Wang et al., 2024a) have made significant strides by leveraging UAV simulators to mitigate the scarcity of aerial VLN datasets, thereby driving advances in this field. However, several critical challenges remain to be addressed:

- **Limited data diversity.** Existing methods rely on AirSim and Unreal Engine (UE) for UAV control, which confines them to digital assets compatible with these platforms, limiting the diversity of available data and constraining the incorporation of more photorealistic sources.
- **High collection cost.** The process of generating trajectories relies on pilots operating UAVs in simulators, followed by manual annotation to create language instructions. The entire process is labor-intensive, time-consuming, and difficult to scale.
- **Small data scale.** Current datasets for aerial VLN remain relatively small, containing only about 10k trajectories, which is far behind embodied manipulation datasets. By contrast, Open X-Embodiment (O'Neill et al., 2024) and EO-1 (Qu et al., 2025) have collected over 1M episodes of manipulation, significantly promoting the development of vision-language-action (VLA) models.

To address these issues, we propose **OpenFly**, a comprehensive platform consisting of diverse rendering engines, a versatile toolchain, and a large-scale benchmark for the aerial VLN task. **To enhance data diversity**, the platform is established on various widely-used rendering engines and advanced techniques, *i.e.,* UE, GTA V, Google Earth, and 3D Gaussian Splatting (3D GS), enabling us to utilize a wide range of assets as shown in Fig. 1. In particular, we use UAVs to capture numerous real-world images and integrate 3D GS technology into our platform to reconstruct realistic 3D scenes, empowering real-to-sim simulation. **To improve the efficiency of data collection**, we develop a versatile toolchain for automated aerial VLN data generation as depicted in Fig. 2. Specifically, point cloud acquisition is first conducted to capture the 3D occupancy of a scene. Next, scene semantic segmentation is performed to identify and select landmarks as waypoints along the flight trajectories. Building on these tools, trajectory generation is then carried out, taking landmarks and point clouds as input, using predefined flight actions as basic units, and automatically searching for a collision-free trajectory. Finally, we feed the trajectories and corresponding UAV-egocentric

images into a vision-language-model (VLM), *e.g.,* GPT-4o, to generate linguistic instructions. The entire pipeline is highly automated, reducing the reliance on UAV pilots and annotators. **To collect a large-scale dataset**, we meticulously collected 18 high-quality scenes, generating various trajectories of differing heights and lengths. Benefitting from our toolchain, we are able to quickly construct a dataset of **100k** samples, significantly larger than existing datasets.

Besides, we propose **OpenFly-Agent**, a keyframe-aware aerial VLN model incorporating an adaptive frame-level sampling mechanism to emphasize critical observations containing instruction-related landmarks, leading to performance improvement and computation reduction compared to a uniform sampling strategy. Extensive experiments are conducted on the OpenFly dataset to evaluate numerous methods, establishing a comprehensive benchmark for the aerial VLN tasks. Overall, our contributions can be summarized as follows:

- We build OpenFly on multiple rendering engines and develop a versatile toolchain, enabling the automatic generation of data with high diversity and efficiency.
- We have constructed a large-scale aerial VLN benchmark comprising 100k trajectories across 18 high-quality scenes. To the best of our knowledge, this is the largest aerial VLN benchmark to date, and users can collect more customized data using the OpenFly platform.
- We propose OpenFly-Agent, a keyframe-aware VLN model. Extensive experiments in both simulated and real-world settings demonstrate its superior performance.

## 2 Related Works

### 2.1 Vision-Language Navigation Datasets

Numerous datasets have been constructed to accelerate the VLN task. R2R (Anderson et al., 2018) focuses on evaluating agents in unseen buildings and provides discrete navigation options. RxR (Ku et al., 2020) provides a more densely annotated VLN dataset. TouchDown (Chen et al., 2019) and REVERIE (Qi et al., 2020) have each contributed a dataset from real-life environments, which requires a ground-based agent to follow instructions and find a target object. CVDN (Thomason et al., 2019) presents a cooperative VLN dataset where agents can access the history of human cooperation for inference. All the above datasets are graph-based, where navigable points are predefined. LANI (Misra et al., 2018) and VLN-CE (Krantz et al., 2020) propose the VLN task in continuous outdoor/indoor environments, enabling agents to move freely to any unobstructed point. Recently, a few works have tried to construct VLN datasets for aerial space. ANDH (Fan et al., 2022) establishes a dialogue-based aerial VLN dataset with bird-view images. CityNav (Lee et al., 2024) builds on the point cloud data from SensatUrban (Hu et al., 2022) and linguistic annotations from CityRefer (Miyanishi et al., 2023), which requires a real-world 2D map to help locate specific landmarks in the instruction. AerialVLN (Liu et al., 2023), OpenUAV (Wang et al., 2024a) and CityNav-Agent (Zhang et al., 2025) integrate AirSim and UE to create VLN scenes where pilots can control UAVs to generate various trajectories.

### 2.2 Vision-Language Navigation Methods

VLN methods enable agents to follow language instructions based on visual observations. Early approaches, such as graph-based methods (Ma et al., 2019; Wang et al., 2019; Ke et al., 2019; Fu et al., 2020), model the environment as a set of predefined nodes, with agents navigating between these discrete states. However, these methods are limited in dynamic, real-world environments. In recent years, LLM-driven approaches (Zhou et al., 2024b;a; Chen et al., 2024; Zeng et al., 2025) have utilized large language models to enhance reasoning and infer navigation steps, offering more flexibility in continuous environments. Despite significant progress, LLM-based methods still face challenges in grounding language instructions with real-world sensory data and adapting to unknown environments. Meanwhile, training-free LLM-based methods (Hu et al., 2025; Xu et al., 2025) provide a flexible way to infer navigation steps from language alone, enabling rapid adaptation of agents without retraining. In contrast, works like (Irshad et al., 2021; Krantz et al., 2021; Zhang et al., 2024; Song et al., 2025) have shifted focus to continuous spaces, aiming for more realistic navigation in

dynamic settings. More recently, aerial VLN has gained attention, with AerialVLN (Liu et al., 2023) proposing a lookahead guidance method for better training trajectories, while STMR (Gao et al., 2024) enhances spatial reasoning through matrix representations, and OpenUAV (Wang et al., 2024a) integrates human feedback with ground-truth trajectories to guide navigation.

# 3 Automatic Data Generation

In this section, we first introduce the rendering engines and data resources, then present the developed toolchain. The overall framework for automatic data generation is illustrated in Fig. 2.

## 3.1 Rendering Engines and Data Resources

We leverage multiple rendering engines to construct diverse and realistic environments. Specifically, **Unreal Engine** provides eight urban scenes spanning over $100km^2$ with rich assets such as buildings, vehicles, and pedestrians. **GTA V** contributes a highly realistic cityscape modeled after Los Angeles. **Google Earth** offers four urban regions (Berkeley, Osaka, Washington D.C., and St. Louis) covering $53.60km^2$. Besides, hierarchical **3D Gaussian Splatting** (Kerbl et al., 2024) is employed for the reconstruction of real-world environments from UAV data, encompassing more than $7km^2$ across five campuses with diverse landmarks. More details and examples are provided in Appendix A.

## 3.2 Toolchain for Automatic Data Collection

To achieve automatic data generation, we first integrate the above rendering engines and design three unified interfaces to control the agent movement and acquire sensor data (presented in Appendix C). Based on these interfaces, we further develop a toolchain, streamlining point cloud acquisition, scene semantic segmentation, trajectory creation, and instruction generation.

**3D Point Cloud Acquisition.** OpenFly integrates various rendering engines and scenes, exhibiting distinct characteristics. To address these differences, we provide two methods to reconstruct the point cloud map for different scenes. 1) Rasterized Sampling Reconstruction: For UE and GTA V scenes, we customize rasterized sampling points at appropriate resolutions, followed by using the developed interface to obtain the local point cloud at the sampling points and stitch them for the entire scene. 2) Image-Based Sparse Reconstruction: In 3D GS, the scene reconstruction process begins with the open-source COLMAP (Schönberger & Frahm, 2016) framework, which generates a sparse point cloud from input images. We directly export and use the point clouds from this step.

**Scene Semantic Segmentation.** VLN requires meaningful landmarks as navigation targets. Thus, we offer three semantic segmentation methods to identify landmarks. 1) 3D Scene Understanding: A sequence of top-down views of the scene is captured in a rasterized format, followed by the off-the-shelf Octree-Graph (Wang et al., 2024b) to extract semantic 3D instances. 2) Point Cloud Projection and Contour Extraction: We acquire the point cloud of a scene and project the voxelized point cloud onto the ground. For each instance, its contour is segmented, and the maximum height of its points is used as the final height. Additionally, semantic annotations are obtained by feeding the segmented instances to GPT-4o for caption. 3) Manual Annotation: When the point cloud quality of a scene is low or finer segmentation is required, OpenFly provides an interface for manually annotating instances and semantics within the point cloud. Users can choose these methods flexibly based on their requirements. The corresponding details and results are shown in Appendix D.

**Automatic Trajectory Generation.** Leveraging the point cloud map and segmentation tools, OpenFly can automatically generate VLN trajectories using the following method. First, a global voxel map $M_{global}$ is constructed from the scene point cloud. Second, a landmark is randomly chosen as the target, with a starting point being selected at a certain distance from the landmark, and a point close to the landmark being chosen as the endpoint. Third, A collision-free trajectory is generated using the A* (Hart et al., 1968) pathfinding algorithm based on $M_{global}$ and a customized action space. By repeatedly selecting the endpoint as the new starting point, complex trajectories can be generated. Finally, utilizing OpenFly's interface, UAV-egocentric images corresponding to the trajectory points are obtained as visual observations. More details are included in Appendix E.

Table 1: Comparisons of different VLN datasets. $N_{traj}$: the number of total trajectories. $N_{vocab}$: vocabulary size. Path Len: the average length of trajectories, measured in meters. Intr Len: the average length of instructions. $N_{act}$: the average number of actions per trajectory.

| Dataset | $N_{traj}$ | $N_{vocab}$ | Path Len. | Intr Len. | Action Space | $N_{act}$ | Environment |
|---|---|---|---|---|---|---|---|
| R2R (Anderson et al., 2018) | 7189 | 3.1K | 10.0 | 29 | graph-based | 5 | Matterport3D |
| RxR (Ku et al., 2020) | 13992 | 7.0K | 14.9 | 129 | graph-based | 8 | Matterport3D |
| TouchDown (Chen et al., 2019) | 9326 | 5.0K | 313.9 | 90 | graph-based | 35 | Google Street View |
| VLN-CE (Krantz et al., 2020) | 4475 | 4.3K | 11.1 | 19 | 2 DoF | 56 | Matterport3D |
| AerialVLN (Liu et al., 2023) | 8446 | 4.5K | 661.8 | 83 | 4 DoF | 204 | AirSim + UE |
| CityNav (Lee et al., 2024) | 32637 | 6.6K | 545 | 26 | 4 DoF | - | SensatUrban |
| OpenUAV (Wang et al., 2024a) | 12149 | 10.8K | 255 | 104 | 6 DoF | 264 | AirSim + UE |
| Ours | 100K | 15.6K | 99.1 | 59 | 4 DoF | 35 | AirSim + UE, GTA V, Google Earth Studio, 3D GS + SIBR viewers |

**Automatic Instruction Generation.** Most previous works have predominantly relied on manual annotation to generate trajectory instructions, which is costly and hinders dataset scalability (Liu et al., 2023; Lee et al., 2024). To address this issue, we propose a highly automated instruction generation method based on VLMs, *e.g.,* GPT-4o.

A straightforward method would be to input all images to VLMs to analyze the trajectory and generate instructions. However, using all images introduces considerable computational overhead and causes significant difficulties for a VLM to understand. Additionally, we find the 'Forward' action usually occupies a larger proportion of a flight trajectory, with 'Turn Left/Turn Right' or 'Ascend/Descend' actions taken when encountering key landmarks. Therefore, we split the complete trajectory into multiple sub-trajectories according to action transitions, extracting key actions and images for processing. Notably, slight angle adjustments often occur during flight to change direction subtly, which will be ignored in this procedure. We submit the action sequence and the last captured three images of each sub-trajectory to a VLM to generate a sub-instruction of both the action and the landmark. All sub-instructions of the same trajectory are then processed by an LLM to integrate into a complete instruction. The proposed strategy significantly improves the instruction accuracy compared to directly inputting all trajectory images to a VLM. To further verify the data quality, we randomly select 3K samples from the entire dataset according to the data distribution in Sec. 4.2. After manually inspecting these samples, we find that they reach a high qualification rate of 91%. The problematic data involves some vague descriptions, but it is still considered acceptable by examiners. Besides, all test data have undergone manual inspection, with low-quality ones removed. Thanks to GPT's high concurrency, we can quickly generate a large number of instructions, which solves the problem of difficult and time-consuming manual annotation. **More details of instruction generation and data quality control are provided in Appendix F and G.**

## 4 Dataset Analysis

Table 1 summarizes key statistics of several commonly used VLN datasets, from which we can see that our dataset features a significantly larger number of trajectories and a greater environmental diversity. In contrast, our average trajectory length and instruction length are relatively short. This is intentional, we argue that short- and medium-range instructions better reflect natural human usage habits and may be more beneficial for advancing aerial VLN.

### 4.1 Trajectory and Instruction Analysis

Using our toolchain, we collect a dataset of 100K trajectories, which is much larger than other aerial VLN datasets. Compared with ground-based VLN, the aerial VLN task has more motion dimensions. Therefore, we set different trajectory lengths and flight heights to obtain rich data. Fig. 3a and 3b exhibit the distribution of these data, with their lengths ranging from 0 to 300 meters, and the heights ranging from 0 to 210 meters. Notably, we follow the mainstream methods (Krantz et al., 2020; Liu et al., 2023) to use discrete actions, *e.g.,* 'Forward' and 'Turn left', for trajectory generation, where the step size of the 'Forward' action is set to 3 m, 6 m, and 9 m to adapt to targets at different distances. Fig. 3c presents the action distribution of our dataset. It should

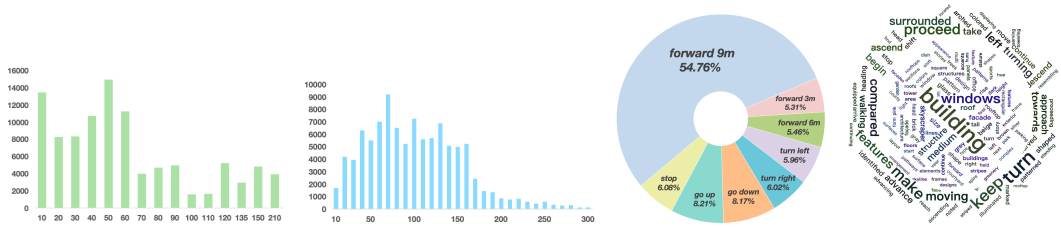



Figure 3: Statistical analysis of the generated data. (a) Height distributions of trajectories. (b) Length distributions of trajectories. (c) Action distributions. (d) Word cloud of verbs and nouns.

be noted that collected trajectories also provide corresponding waypoint information, which can be further processed into smoother trajectories to enable navigation waypoint prediction. Besides, the OpenFly platform supports trajectory generation with continuous waypoints directly based on drone trajectory planning algorithms Wang et al. (2022); Zhou et al. (2021); Mellinger & Kumar (2011); Zhou et al. (2019). To further enhance data diversity, we incorporate the DAgger Ross et al. (2011) algorithm as a data augmentation functionality. In summary, OpenFly offers a comprehensive platform that allows users to generate custom data on their own. It also supports agent interaction and enables real-time retrieval of both agent poses and environmental data. This makes it compatible with On-policy training approaches.

For instruction analysis, the vocabulary size of our dataset is 15.6K, and the average length of instructions is 59. Fig. 3d illustrates the word clouds of nouns and verbs, where 'building', 'windows', and 'skyscraper' are the most common references, and 'proceed' and 'turn' are the mostly used verbs for VLN. Due to the space limitation, **we put more details in Appendix H.**

### 4.2 Dataset Split

Similar to previous works, we divide the dataset into three splits, *i.e., Train, Test Seen, Test Unseen*. For the *Train* split, 7 scenes under the UE rendering engine account for 75.7% of all data, since UE provides the largest number of scenes, where different amounts of trajectories are sampled according to the areas of scenes. The 4 scenes created by 3D GS are also the main part of the data, accounting for nearly 20% of the total amount. To ensure visual quality, we only collect data from a high-altitude perspective using Google Earth, which accounts for 4.46%. The *Test Seen* data consists of 1800 trajectories uniformly sampled from 11 seen scenarios, and the *Test Unseen* data comprises 1200 trajectories uniformly generated from 3 unseen scenes, *i.e.,* UE-smallcity, 3D GS-sjtu02, and a Los Angeles-like city in GTA V. Detailed data distributions are shown in Appendix H.

## 5 OpenFly-Agent

Fig. 4 illustrates the architecture of our OpenFly-Agent, an aerial VLN model that builds upon the OpenVLA (Kim et al., 2024) baseline, since OpenVLA and aerial VLN share a similar pipeline, *i.e.*, taking images and instructions as input and generating actions. OpenVLA is trained on 1M data, having strong abilities in instruction-following and reasoning, which establishes an efficient initialization for our model. In contrast, our OpenFly-Agent takes a sequence of images as input to indicate the observation history instead of one image in the original OpenVLA. Additionally, to mitigate visual redundancy between adjacent video frames while maintaining key information, two strategies are designed, *i.e.,* keyframe selection and visual token merging. First, a series of candidate keyframes is selected based on the UAV flight trend and a landmark grounding module. Then, these keyframes are merged temporally, resulting in a compact sequence of visual tokens. Finally, the action decoder discretizes the predicted tokens to 6 action types specific to UAVs.

### 5.1 Keyframe Selection

The length of contextual visual tokens is a major challenge for VLMs when processing videos. Many open-source VLMs use uniform frame sampling (Buch et al., 2022; Ranasinghe et al., 2024; Wang

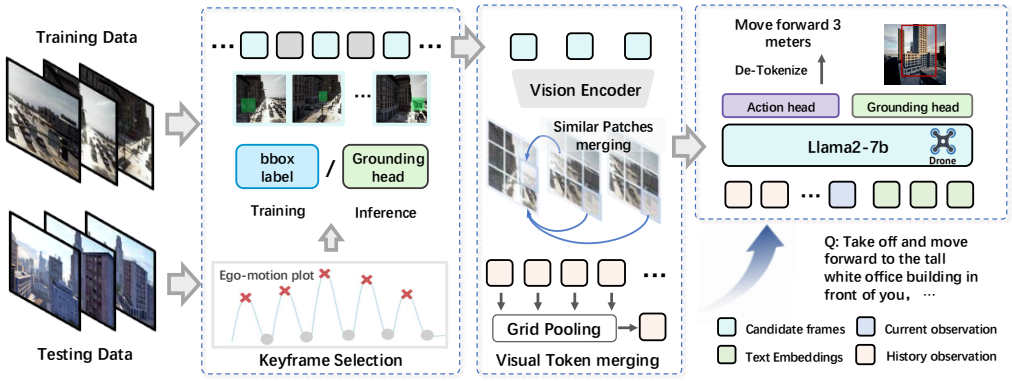



Figure 4: The architecture of OpenFly-Agent. Keyframes are selected according to action transitions and the landmark grounding module to extract crucial observations as the history, with corresponding visual tokens compressed to further reduce the computational burden.

et al., 2025) to reduce calculation, but this strategy is not suitable for aerial VLN, since it may miss frames containing key landmarks. To address this issue, a keyframe selection strategy is proposed to emphasize important visual observations. We notice that sudden changes in the UAV's trajectory are often caused by the observation of landmarks, which can serve as a kind of cues to determine keyframes. Therefore, a heuristic method is adopted to select candidate frames by identifying the change point of the UAV's movement, followed by extracting the corresponding frame and two frames before and after it from the trajectory, constituting a keyframe set. Moreover, we design a landmark grounding module, which consists of three cross-attention layers to incorporate text and image features from the LLM hidden state, predicting the bounding boxes $\mathrm{b} \in \mathcal{R}^4$ of the instruction-indicated landmark. To incorporate as many landmark-related regions as possible into the historical visual tokens, candidate frames with the bounding boxes' area greater than the threshold $\theta$ will be retained as the final keyframes. During the training process, we obtain the bounding box of each landmark using the developed tools introduced in Sec. 3.2, enabling the training of the grounding module and the accurate selection of keyframes. During the testing process, the bounding box of each frame is sequentially estimated by the well-trained grounding module. Then, our model selects keyframes by bounding boxes area and adjacent frames when a significant motion change occurs, forming a keyframe set for this moment.

### 5.2 Visual Token Pruning

To further reduce redundant information in keyframes, we introduce visual token merging into OpenFly-Agent. For the keyframes selected by the above method, a visual encoder maps them to multiple visual tokens, with each token representing the information of an image patch. Considering the potential inter-frame patch redundancy, we take a strategy that similar tokens in adjacent frames are periodically merged. Specifically, we select the frame with the largest bounding box in a keyframe set as the reference, since it usually contains the crucial observation indicating the landmark in an instruction. Then, we densely calculate the cosine similarities between each pair of visual tokens of the reference image and other comparative images in a keyframe set. Next, we merge the tokens with high similarity by averaging them, with the unmerged tokens in the comparative frame being discarded. The merging operation is iteratively performed until the entire keyframe set has been traversed. Besides, we maintain a memory bank with a capacity of $K$ images, following a first-in-first-out (FIFO) policy to retain the latest keyframes. Since aerial VLN requires UAVs to perform long-distance flights based on instructions, we continue to conduct token compression within each keyframe to reduce the computational burden. The compressed visual tokens are obtained through grid pooling (Li et al., 2024). Notably, we keep the visual tokens of the current frame uncompressed to capture the latest visual observation, as it contains the most important information for action prediction.

Table 2: Comparison results on the test set. 'Random' means randomly selecting one action to execute until the 'stop' action is chosen. All models are retrained using our dataset.

| Method | test-seen | | | | test-unseen | | | |
|---|---|---|---|---|---|---|---|---|
| | NE↓ | SR↑ | OSR↑ | SPL↑ | NE↓ | SR↑ | OSR↑ | SPL↑ |
| Random | 242m | 0.7% | 0.8% | 0% | 301m | 0.1% | 0.1% | 0% |
| Seq2Seq (Krantz et al., 2020) | 205m | 2.9% | 24.3% | 2.6% | 229m | 2.1% | 20.6% | 1.1% |
| CMA (Krantz et al., 2020) | 161m | 5.4% | 28.1% | 4.8% | 217m | 4.6% | 24.4% | 2.1% |
| See-Point-Fly (Hu et al., 2025) | - | - | - | - | 191m | 8.2% | 12.7% | 6.3% |
| AerialVLN (Liu et al., 2023) | 139m | 7.5% | 30.0% | 6.8% | 214m | 7.3% | 28.1% | 4.4% |
| Navid (Zhang et al., 2024) | 153m | 13.0% | 38.2% | 11.6% | 210m | 10.8% | 27.2% | 5.0% |
| NaVila (Cheng et al., 2024) | 132m | 20.3% | 53.5% | 17.8% | 202m | 14.7% | 42.1% | 9.6% |
| OpenFly-Agent (Ours) | **93m** | **34.3%** | **64.3%** | **24.9%** | **154m** | **22.6%** | **56.2%** | **19.1%** |

# 6 Experiments

## 6.1 Implementation and Training Details

The proposed OpenFly-Agent adopts the OpenVLA (Kim et al., 2024) as the baseline, with the current frame during flight remaining 256 tokens and all historical keyframes compressed into 1 token. The capacity $K$ of the history memory bank is set to 2 in our experiment. For the action head, the last 256 tokens in the vocabulary are used as special tokens for action representation. Similar to (Liu et al., 2023; Lee et al., 2024), 6 actions for UAVs are defined as {Forward, Turn Left, Turn Right, Move Up, Move Down, Stop}. The OpenFly-Agent is trained with a batch size of 64 and a learning rate of 2e-5. The grounding module is optimized with a GIoU loss function, and the threshold $\theta$ for keyframe selection is set to 0.25 times the size of the input image.

## 6.2 Evaluation Metrics

Four standard metrics in VLN tasks are adopted to evaluate different methods, *i.e.,* navigation error (NE), success rate (SR), oracle success rate (OSR), and success weighted by path length (SPL). NE measures the average deviation between the UAV's final stopping point and the ground-truth destination. SR calculates the proportion of successful tasks, where a task is considered successful if the UAV stops within 20 m of the target (Liu et al., 2023). Each environment provides corresponding point clouds that enable collision checking. If a collision occurs, the task is counted as a failure. In OSR, if any point on the trajectory is within 20 m of the target, the task can be considered successful. SPL calculates the success rate weighted by the ratio of the ground-truth path length to the actually-executed path length.

## 6.3 Quantitative Results

We evaluate the proposed OpenFly-Agent and multiple VLN methods on the test set, with quantitative results listed in Table 2, where Seq2Seq, CMA, and AerialVLN achieve limited success rates. In contrast, Navid (Zhang et al., 2024) and NaVila (Cheng et al., 2024) are two most recent VLN methods, obtaining better results and demonstrating the great potential of VLMs in aerial VLN. See-Point-Fly Hu et al. (2025) is a zero-shot method, which is evaluated using GPT-4.1 as the agent and demonstrates reasonable robustness. Our OpenFly-Agent outperforms the comparison methods by a large margin, benefiting from the proposed strategies. While aerial VLN is an emerging and challenging task, and there is still much room for improvement. The results on the test-unseen split indicate the generalization abilities of these methods. Similarly, our method achieves the best performance, exhibiting a certain degree of robustness. However, all methods are significantly degraded, indicating that more powerful models are urgently needed to be developed.

## 6.4 Real-world Experiments

The real-world experiments are conducted in 23 real outdoor scenes, where each scene corresponds to an unseen VLN task created by human operators, and the trajectory lengths range from 50m to 500m. We use a Q250 airframe as a real agent, carrying an NVIDIA Jetson Xavier NX running

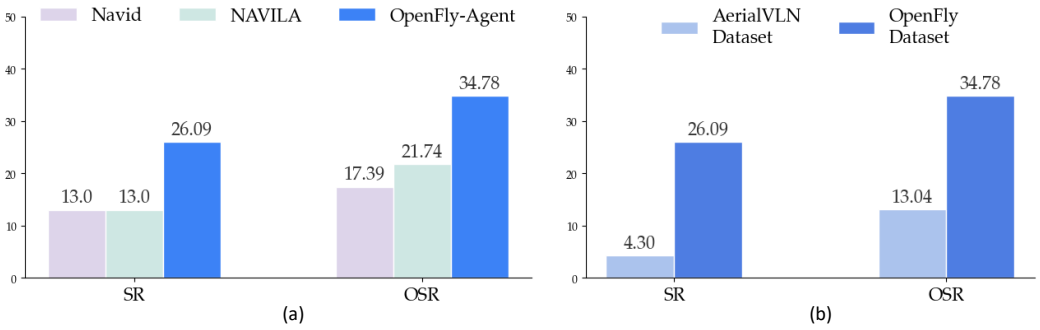



Figure 5: Results of real-world experiments. (a) Comparison with two strong VLN methods. (b) Performances of OpenFly-Agent trained on different datasets.

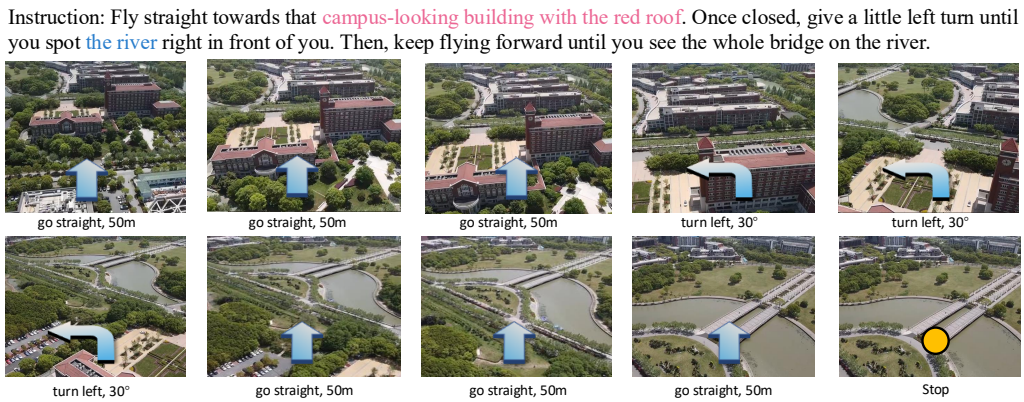



Figure 6: Snapshots of the real-world experiment.

Ubuntu 18.04 as the onboard computer. In the real-world experiments with the drone, we utilize the "Super" (Ren et al., 2025) trajectory planning framework for local trajectory planning and employ Model Predictive Control (MPC) Falanga et al. (2018) for trajectory tracking. The advantage of this paradigm is that it enables the VLN model to adapt to various planning and control algorithms, thereby accommodating diverse robotic platforms and scenarios. All methods run on an external PC communicating with the onboard computer to transfer images and action instructions. Two most recent models, Navid (Zhang et al., 2024) and NaVila (Cheng et al., 2024), are evaluated for comparison. The results are shown in Fig 5 (a), where our model achieves the best performance with 26.09% SR and 34.78% OSR, significantly outperforming the comparison methods. This experiment again indicates the superiority of our OpenFly-Agent. Besides, we also trained our model on both our own dataset and the AerialVLN dataset separately. The results are shown in Fig. 5 (b), strongly demonstrating the capability of our data generation method in bridging the sim-to-real gap. A qualitative result is presented in Fig. 6, and a dynamic demo can be found in our supplementary video.

### 6.5 Ablation Study

Ablation studies are conducted to evaluate the contribution of the keyframe selection and visual token merging in OpenFly-Agent. Table. 3 shows the results, where OpenVLA (Kim et al., 2024) is our baseline. Using only the current frame or uniformly selecting from previous observation as keyframes makes the model perform poorly in the aerial VLN task. From 'History + VTM', we can see that historical frames significantly improve the success rate. The keyframe selection strategy further increases the SR from 16.6% to 34.3%, demonstrating the effectiveness of key observations. Besides, the comparison between 'KS' and 'KS + VTM' indicates the great effect of our visual token merging strategy. We find that there is a severe imbalance between the number of text and image tokens if the token merging strategy is not applied. The cross-modal signal can be diluted

by the numerical imbalance between a few text tokens and many visual tokens Luo et al. (2025). As a result, background clutter, task-irrelevant distractors, and environmental noise may be encoded indiscriminately, leading to excessive computational cost and diluted attention to task critical cues Li et al. (2026).

## 7 Conclusion

In this work, we present OpenFly, a platform designed for large-scale data collection in aerial Vision-and-Language Navigation (VLN). OpenFly integrates multiple rendering engines and provides an automatic toolchain for data generation, enabling efficient collection of diverse, high-quality aerial VLN data. The resulting large-scale dataset comprises 100k trajectories across 18 distinct scenes, spanning a wide range of altitudes and lengths, which is significantly larger than existing ones. Furthermore, we propose OpenFly-Agent, a keyframe-aware aerial VLN model capable of identifying frames with critical observations, leading to accurate flight action prediction. Extensive experiments validate the effectiveness of the proposed method, and establish a comprehensive benchmark for future advancements in aerial VLN.

Table 3: Ablation study on the test-seen split. 'KS' and 'VTM' denote keyframe selection and visual token merging, respectively. 'History' indicates uniform frame sampling. 'Random KS' means randomly selecting a frame from the candidate keyframe set.

| Method | NE↓ | SR↑ | OSR↑ | SPL↑ |
|---|---|---|---|---|
| OpenVLA (baseline) | 231m | 2.3% | 10.8% | 2.2% |
| History | 223m | 6.9% | 23.3% | 5.6% |
| Random KS | 264m | 8.7% | 26.6% | 5.8% |
| KS | 275m | 9.2% | 28.1% | 6.1% |
| History + VTM | 215m | 16.6% | 40.5% | 9.1% |
| KS + VTM | 93m | 34.3% | 64.3% | 24.9% |

## References


Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian D. Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2, 3, 5

Steven Bird. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pp. 69–72, 2006. 19

Sebastien Bonopera, Peter Hedman, Jerome Esnault, Siddhant Prakash, Simon Rodriguez, Theo Thonat, Mehdi Benadel, Gaurav Chaurasia, Julien Philip, and George Drettakis. sibr: A system for image based rendering, 2020. 16

Shyamal Buch, Cristóbal Eyzaguirre, Adrien Gaidon, Jiajun Wu, Li Fei-Fei, and Juan Carlos Niebles. Revisiting the" video" in video-language understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2917–2927, 2022. 6

Yihan Cao, Jiazhao Zhang, Zhinan Yu, Shuzhen Liu, Zheng Qin, Qin Zou, Bo Du, and Kai Xu. Cognav: Cognitive process modeling for object goal navigation with llms. *ICCV*, 2025. 2

Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. TOUCHDOWN: natural language navigation and spatial reasoning in visual street environments. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 2, 3, 5

Jiaqi Chen, Bingqian Lin, Ran Xu, Zhenhua Chai, Xiaodan Liang, and Kwan-Yee Kenneth Wong. Mapgpt: Map-guided prompting with adaptive path planning for vision-and-language navigation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 9796–9810, 2024. 3

Peihao Chen, Dongyu Ji, Kunyang Lin, Runhao Zeng, Thomas H Li, Mingkui Tan, and Chuang Gan. Weakly-supervised multi-granularity map learning for vision-and-language navigation. *arXiv preprint arXiv:2210.07506*, 2022. 2

An-Chieh Cheng, Yandong Ji, Zhaojing Yang, Zaitian Gongye, Xueyan Zou, Jan Kautz, Erdem Bıyık, Hongxu Yin, Sifei Liu, and Xiaolong Wang. Navila: Legged robot vision-language-action model for navigation. *arXiv preprint arXiv:2412.04453*, 2024. 8, 9

Davide Falanga, Philipp Foehn, Peng Lu, and Davide Scaramuzza. Pampc: Perception-aware model predictive control for quadrotors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–8. IEEE, 2018. 9

Yue Fan, Winson X. Chen, Tongzhou Jiang, Chun ni Zhou, Yi Zhang, and Xin Wang. Aerial vision-and-dialog navigation. *ArXiv*, abs/2205.12219, 2022. 3, 19

Tsu-Jui Fu, Xin Eric Wang, Matthew F. Peterson, Scott T. Grafton, Miguel P. Eckstein, and William Yang Wang. *Counterfactual Vision-and-Language Navigation via Adversarial Path Sampler*, pp. 71–86. 2020. 3

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. *CoRR*, abs/2410.08500, 2024. 4

Wenxuan Guo, Xiuwei Xu, Hang Yin, Ziwei Wang, Jianjiang Feng, Jie Zhou, and Jiwen Lu. Igl-nav: Incremental 3d gaussian localization for image-goal navigation. *ICCV*, 2025. 2

Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. 4, 18

Yicong Hong, Zun Wang, Qi Wu, and Stephen Gould. Bridging the gap between learning in discrete and continuous environments for vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15439–15449, 2022. 2

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, 2025. URL https://arxiv.org/abs/2509.22653. 3, 8

Qingyong Hu, Bo Yang, Sheikh Khalid, Wen Xiao, Niki Trigoni, and Andrew Markham. Sensaturban: Learning semantics from urban-scale photogrammetric point clouds. *Int. J. Comput. Vis.*, 130(2):316–343, 2022. 3

MuhammadZubair Irshad, NiluthpolChowdhury Mithun, Zachary Seymour, Han-Pang Chiu, Supun Samarasekera, and Rakesh Kumar. Sasra: Semantically-aware spatio-temporal reasoning agent for vision-and-language navigation in continuous environments. *arXiv: Robotics,arXiv: Robotics*, 2021. 3

Liyiming Ke, Xiujun Li, Yonatan Bisk, Ari Holtzman, Zhe Gan, Jingjing Liu, Jianfeng Gao, Yejin Choi, and Siddhartha Srinivasa. Tactical rewind: Self-correction via backtracking in vision-and-language navigation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A hierarchical 3d gaussian representation for real-time rendering of very large datasets. *ACM Transactions on Graphics (TOG)*, 43(4):1–15, 2024. 4, 16

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. 6, 8, 9

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 - 16th European Conference, Glasgow*, 2020. 2, 3, 5, 8

Jacob Krantz, Aaron Gokaslan, Dhruv Batra, Stefan Lee, and Oleksandr Maksymets. Waypoint models for instruction-guided navigation in continuous environments. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 3

Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020. 2, 3, 5

Jungdae Lee, Taiki Miyanishi, Shuhei Kurita, Koya Sakamoto, Daichi Azuma, Yutaka Matsuo, and Nakamasa Inoue. Citynav: Language-goal aerial navigation dataset with geographic information. *CoRR*, abs/2406.14240, 2024. 3, 5, 8

Wei Li, Renshan Zhang, Rui Shao, Zhijian Fang, Kaiwen Zhou, Zhuotao Tian, and Liqiang Nie. Semanticvla: Semantic-aligned sparsification and enhancement for efficient robotic manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026. 10

Yanwei Li, Chengyao Wang, and Jiaya Jia. Llama-vid: An image is worth 2 tokens in large language models. In *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part XLVI*, volume 15104, pp. 323–340, 2024. 7

Shubo Liu, Hongsheng Zhang, Yuankai Qi, Peng Wang, Yanning Zhang, and Qi Wu. Aerialvln: Vision-and-language navigation for uavs. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pp. 15338–15348, 2023. 2, 3, 4, 5, 8, 15, 19

Yuxing Long, Wenzhe Cai, Hongcheng Wang, Guanqi Zhan, and Hao Dong. Instructnav: Zero-shot system for generic instruction navigation in unexplored environment. *arXiv preprint arXiv:2406.04882*, 2024. 2

Jiabin Luo, Junhui Lin, Zeyu Zhang, Biao Wu, Meng Fang, Ling Chen, and Hao Tang. Univid: The open-source unified video model. *arXiv preprint arXiv:2509.24200*, 2025. 10

Chih-Yao Ma, Jing Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. Self-monitoring navigation agent via auxiliary progress estimation. *arXiv: Artificial Intelligence,arXiv: Artificial Intelligence*, 2019. 3

Tianyi Ma, Yue Zhang, Zehao Wang, and Parisa Kordjamshidi. Breaking down and building up: Mixture of skill-based vision-and-language navigation agents, 2025. 2

Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9-13 May 2011*, pp. 2520–2525, 2011. 6

Dipendra Kumar Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi. Mapping instructions to actions in 3d environments with visual goal prediction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels*, 2018. 2, 3

Taiki Miyanishi, Fumiya Kitamori, Shuhei Kurita, Jungdae Lee, Motoaki Kawanabe, and Nakamasa Inoue. Cityrefer: Geography-aware 3d visual grounding dataset on city-scale point cloud data. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. 3

Abby O'Neill, Abdul Rehman, and et al. Open x-embodiment: Robotic learning datasets and RT-X models : Open x-embodiment collaboration. In *IEEE International Conference on Robotics and Automation, ICRA 2024, Yokohama, Japan, May 13-17, 2024*, pp. 6892–6903, 2024. 2

Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. REVERIE: remote embodied visual referring expression in real indoor environments. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, 2020. 2, 3, 19

Delin Qu, Haoming Song, Qizhi Chen, Zhaoqing Chen, Xianqiang Gao, Xinyi Ye, Qi Lv, Modi Shi, Guanghui Ren, Cheng Ruan, Maoqing Yao, Haoran Yang, Jiacheng Bao, Bin Zhao, and Dong Wang. Eo-1: Interleaved vision-text-action pretraining for general robot control. *arXiv preprint*, 2025. URL https://arxiv.org/abs/2508.21112. 2

Kanchana Ranasinghe, Xiang Li, Kumara Kahatapitiya, and Michael S Ryoo. Understanding long videos in one multimodal language model pass. *arXiv preprint arXiv:2403.16998*, 2024. 6

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. 9

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, volume 15 of *JMLR Proceedings*, pp. 627–635, 2011. 6

Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4, 17

Xinshuai Song, Weixing Chen, Yang Liu, Weikai Chen, Guanbin Li, and Liang Lin. Towards long-horizon vision-language navigation: Platform, benchmark and method. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 12078–12088, 2025. 3

Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. Vision-and-dialog navigation. In *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, pp. 394–406, 2019. 2, 3

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. *CoRR*, abs/2410.07087, 2024a. 2, 3, 4, 5

Xiaohan Wang, Yuhui Zhang, Orr Zohar, and Serena Yeung-Levy. Videoagent: Long-form video understanding with large language model as agent. In *European Conference on Computer Vision*, pp. 58–76. Springer, 2025. 6

Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

Zhepei Wang, Xin Zhou, Chao Xu, and Fei Gao. Geometrically constrained trajectory optimization for multicopters. *IEEE Trans. Robotics*, 38(5):3259–3278, 2022. 6

Zhigang Wang, Yifei Su, Chenhui Li, Dong Wang, Yan Huang, Bin Zhao, and Xuelong Li. Open-vocabulary octree-graph for 3d scene understanding. *CoRR*, abs/2411.16253, 2024b. 4, 17

Zihan Wang, Xiangyang Li, Jiahao Yang, Yeqi Liu, and Shuqiang Jiang. Sim-to-real transfer via 3d feature fields for vision-and-language navigation. *arXiv preprint arXiv:2406.09798*, 2024c. 2

Qiu Weichao, Zhong Fangwei, Zhang Yi, Qiao Siyuan, Xiao Zihao, Kim Tae, Wang Yizhou, and Yuille Alan. Unrealcv: Virtual worlds for computer vision. *ACM Multimedia Open Source Software Competition*, 2017. 15

Haotian Xu, Yue Hu, Chen Gao, Zhengqiu Zhu, Yong Zhao, Yong Li, and Quanjun Yin. Geonav: Empowering mllms with explicit geospatial reasoning abilities for language-goal aerial navigation, 2025. URL https://arxiv.org/abs/2504.09587. 3

Shuang Zeng, Xinyuan Chang, Mengwei Xie, Xinran Liu, Yifan Bai, Zheng Pan, Mu Xu, and Xing Wei. Futuresightdrive: Thinking visually with spatio-temporal cot for autonomous driving. *arXiv preprint arXiv:2505.17685*, 2025. 3

Jiazhao Zhang, Kunyu Wang, Rongtao Xu, Gengze Zhou, Yicong Hong, Xiaomeng Fang, Qi Wu, Zhizheng Zhang, and He Wang. Navid: Video-based vlm plans the next step for vision-and-language navigation. *Robotics: Science and Systems*, 2024. 2, 3, 8, 9

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, 2025. URL https://arxiv.org/abs/2505.05622. 3

Boyu Zhou, Fei Gao, Luqi Wang, Chuhao Liu, and Shaojie Shen. Robust and efficient quadrotor trajectory generation for fast autonomous flight. *IEEE Robotics Autom. Lett.*, 4(4):3529–3536, 2019. 6

Gengze Zhou, Yicong Hong, Zun Wang, Xin Eric Wang, and Qi Wu. Navgpt-2: Unleashing navigational reasoning capability for large vision-language models. In *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part VII*, volume 15065, pp. 260–278, 2024a. 3

Gengze Zhou, Yicong Hong, and Qi Wu. Navgpt: Explicit reasoning in vision-and-language navigation with large language models. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pp. 7641–7649, 2024b. 3

Xin Zhou, Zhepei Wang, Hongkai Ye, Chao Xu, and Fei Gao. Ego-planner: An esdf-free gradient-based local planner for quadrotors. *IEEE Robotics Autom. Lett.*, 6(1):478–485, 2021. 6

# APPENDIX

## A MORE DETAILS OF RENDERING ENGINES AND DATA RESOURCES

In this section, more information about the used rendering engines and data resources are detailed, with several high-quality examples illustrated in Fig. 7.


Figure 7: High-quality examples from different rendering engines and techniques, including several large cities such as Shanghai, Guangzhou, Los Angeles, Osaka, and etc., cover an area of over a hundred square kilometers in total. 3D GS provides five large campus scenes, further enhancing the diversity and realism of the data.

**Unreal Engine.** UE is a rendering engine capable of providing highly realistic interactive virtual environments. This platform has undergone five iterations, and each version features comprehensive and high-quality digital assets. In UE5, we meticulously select an official sample project named 'City Sample', which provides us with a large urban scene covering $25.3km^2$ and a smaller one covering $2.7km^2$. These scenes include a variety of assets such as buildings, streets, traffic lights, vehicles, and pedestrians. Besides, in UE4, we prepare six more high-quality scenes. Specifically, there are two large scenes showcasing the central urban areas of Shanghai and Guangzhou, covering areas of $30.88km^2$ and $58.56km^2$, respectively. The remaining four scenes are selected from AerialVLN (Liu et al., 2023). They have smaller areas for totally about $26.64km^2$. These scenes encompass a wide range of architectural styles, including both Chinese and Western influences, as well as classical and modern designs. Additionally, the UE4 engine allows us to make adjustments in scene time to achieve different appearances of scenes under varying lighting conditions.

AirSim is an open-source simulator, which provides highly realistic simulated environments for UAVs and cars. We integrate the AirSim plugin into UE4 to obtain image data easily from the perspective of a UAV. Since AirSim does not support UE5 and stopped updating in 2022, we use the UnrealCV (Weichao et al., 2017) plugin as an alternative for image acquisition in UE5. To realize a highly efficient data collection in simulated scenes, we modify the UE5 project to a C++ project, integrate the UnrealCV plugin, and package executables for multiple systems like Windows and Linux.

**GTA V.** It is an open-world game that is frequently used by computer vision researchers due to its highly realistic and dynamic virtual environment. The game features a meticulously crafted cityscape modeled after Los Angeles, encompassing various buildings and locations such as skyscrapers, gas stations, parks, and plazas, along with dynamic traffic flows and changes in lighting and shadows.

Script Hook V is a third-party library with the interface to GTA V's native script functions. With the help of Script Hook V, we build an efficient and stable interface, which receives the pose information and returns accurate RGB images and lidar data. From the interface, we can control a virtual agent to collect the required data in an arbitrary pose and angle in the game.

Table 4: Different 3D GS Scenes

| Campus Name | Images | Area |
|---|---|---|
| ECUST (Fengxian Campus) | 12008 | $1.06\,\mathrm{km}^2$ |
| NWPU (Youyi Campus) | 4648 | $0.8\,\mathrm{km}^2$ |
| NWPU (Changan Campus) | 23798 | $2.6\,\mathrm{km}^2$ |
| SJTU (Minhang-East Zone) | 20934 | $1.72\,\mathrm{km}^2$ |
| SJTU (Minhang-West Zone) | 9536 | $0.95\,\mathrm{km}^2$ |

**Google Earth.** It is a virtual globe software, which builds a 3D earth model by integrating satellite imagery, aerial photographs, and Geographic Information System (GIS) data. From this engine, we select four urban scenes covering a total area of $53.60km^2$, *i.e.,* Berkeley, primarily consisting of traditional neighborhoods; Osaka, which features a mix of skyscrapers and historic buildings; and two areas with numerous landmarks: Washington, D.C., and St. Louis.

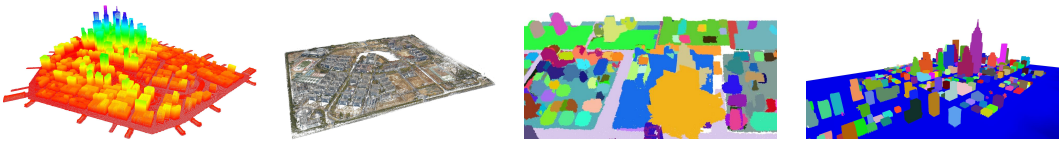
Google Earth Studio is a web-based animation and video production tool that allows us to create keyframes and set camera target points on the 2D and 3D maps of Google Earth. Using this functionality, we can quickly generate customized tour videos by selecting specific routes and angles. In order to efficiently plan the route, we develop a function that automatically draws the flight trajectory in Google Earth Studio according to the selected area and predefined photo interval.

**3D Gaussian Splatting.** As a highly realistic reconstruction method, hierarchical 3D GS (Kerbl et al., 2024) employs a hierarchical training and display architecture, making it particularly suitable for rendering large-scale areas. Due to these features, we use this method to reconstruct and render multiple real scenes. We utilize the DJI M30T drone as the data collection device, which offers an automated oblique photography mode, enabling us to capture a large area of real-world data with minimal manpower. Practically, we gathered data from five campuses across three universities, which are East China University of Science and Technology, Northwestern Polytechnical University, and Shanghai Jiao Tong University (referred to as ECUST, NWPU, and SJTU). These campus scenes include various types and styles of landmarks, such as libraries, bell towers, waterways, lakes, playgrounds, construction sites, and lawns. The detailed information for the five campuses is presented in Table 4.

SIBR (Bonopera et al., 2020) viewers is a rendering tool designed for the 3D GS project, enabling visualization of a scene from arbitrary viewpoints. The tool supports high-frame-rate scene rendering and provides various interactive modes for navigation. Building upon SIBR viewers, we developed an HTTP RESTful API that generates RGB images from arbitrary poses, simulating a UAV's perspective.

## B Details of 3D GS Data Collection

From the UAV's perspective, choosing the appropriate shooting altitude poses a dilemma, *i.e.,* if the altitude is too low, the sparse point cloud generated during the initialization of the 3D GS reconstruction will be suboptimal, due to insufficient feature point matches between photos. In contrast, if the altitude is too high, the Gaussian reconstruction will result in an overly coarse training of details. After multiple attempts, the data collection plan using the M30T was determined as follows. For large-scale block scenes, oblique photography is performed at approximately twice the average building height using the default parameters of the M30T's wide-angle camera, with a tilt angle of -45°. For landmark buildings with heights significantly different from the average height, additional targeted data collection is conducted at twice their height. This altitude setting can, to a certain extent, ensure both higher-quality point cloud initialization and Gaussian splatting training.



(a) (b) (c) (d)

Figure 8: Results of our point cloud acquisition and semantic segmentation. (a) Rasterized sampling point cloud reconstruction. (b) Image-based point cloud reconstruction. (c) Semantic 3D scene segmentation. (d) Point cloud projection and contour extraction.

## C Unified Interfaces

For data collection, we integrate all rendering engines and design three unified interfaces, *i.e.*, the agent movement interface, the lidar data acquisition interface, and the image acquisition interface, allowing an agent to move and perceive the environment within any scene.

- Agent Movement Interface: We design a *CoorTrans* module, which implements a customized pose transformation matrix and scaling function to unify all coordinate systems into a meter-based FLU (Front-Left-Up) convention. This interface enables precise agent positioning, ensuring consistency and facilitating automatic trajectory generation.

- Lidar Data Acquisition Interface: Lidar data is crucial for scene occupancy perception and essential for trajectory generation. Our platform supports different lidar data acquisition methods, including lidar sensor collection, depth map back-projection, and image feature matching. We develop a unified interface to integrate these methods and leverage the proposed *CoorTrans* module to align all data to the same FLU coordinate system.

- Image Acquisition Interface: We integrate HTTP RESTful and TCP/IP protocols to form a unified image request interface, allowing image data to be obtained from any location with flexible resolutions and viewpoints.

## D Results of Point Cloud Acquisition and Semantic Segmentation

This section presents results regarding the point cloud acquisition and scene semantic segmentation, as shown in Fig. 8.

**Point Cloud Acquisition.** OpenFly integrates data resources from different rendering engines. In order to obtain point cloud information from different scenes, we provide two point cloud acquisition methods. 1) For UE and GTAV scenes, we provide a tool that utilizes rasterized point cloud sampling and reconstruction to obtain a global point cloud. The results can be seen in Fig. 8a. 2) For 3D GS scenes, we use COLMAP (Schönberger & Frahm, 2016) to obtain relatively sparse point data from images, as shown in Fig. 8b. Although the point cloud generated by this method is relatively sparse, it provides sufficient coverage information.

**Scene Semantic Segmentation.** To meet the requirements of different data resources and different segmentation granularities, our OpenFly offers three methods for obtaining 3D semantic segmentation of scenes. Here, we present results of segmentation methods other than manual annotation. 1) Fig. 8c illustrates the semantic segmentation results based on the off-the-shelf 3D scene understanding method Octree-Graph (Wang et al., 2024b). This method provides more granular results. 2) The result of semantic segmentation via point cloud projection and contour extraction is shown in Fig. 8d. This method leverages high-precision point clouds to achieve instance segmentation for structures like buildings and trees, which directly contact the ground.

## E A Comprehensive Introduction to Automatic Trajectory Generation

Leveraging the point cloud map and segmentation tools, OpenFly offers two methods for trajectory generation for different scenes. 1) Path search based on customized action space: First, a global voxel map $M_{global}$ and a bird's eye view (BEV) occupancy map $M_{bev}$ are constructed from the scene point cloud. Second, the flight altitude is randomly selected within the user-defined height range, and landmarks that are not lower than the height threshold $H_\tau$ are chosen as targets. A starting point is selected within the distance range $[r, R]$ from the landmark, ensuring that it is not occupied in both $M_{global}$ and $M_{bev}$. Then, a point on the line connecting the starting point and the landmark, which is close to the landmark and unoccupied in $M_{bev}$, is chosen as the endpoint. Third, A collision-free trajectory from the starting point to the endpoint is generated using the A* (Hart et al., 1968) pathfinding algorithm, where the granularity of exploration step size and direction can be adjusted according to the action space. Besides, by repeatedly selecting the endpoint as the new starting point, complex trajectories can be generated. Finally, utilizing OpenFly's interface, images corresponding to the trajectory points can be obtained. 2) Path search based on grid: Google Map data does not allow image retrieval at arbitrary poses in the space. Thus, we rasterize a pre-selected area and collect images from each grid point in all possible orientations. Starting and ending points are chosen within the grid points to generate trajectories. Corresponding images for these trajectory points are then selected from the pre-collected image set.

## F Details of Instruction Generation

Except for the instruction generation described in the main paper. The remaining process is mainly divided into two parts: landmark feature extraction and sub-instruction fusion. A simplified prompt to the VLM and the corresponding response are probably like this.

- Get Landmark features.
  **System Prompt**: You are an assistant who is proficient in image recognition. You can accurately identify the object in the picture and its characteristics that are different from the surrounding objects. I will give you the three final images you will see. Please focus on the last image and tell me the features of the target building and reply to me in the form of JSON.

  **User**: The target is the nearest prominent landmark to me. Answer me a dictionary like color:–, feature: –, size: –, type: –.

  **GPT 4o**: color: blue, feature: Steel, glass, size: medium size, type: building.
- Instruction Fusion.
  **System Prompt**: You are an assistant proficient in text processing. You need to help me combine these scattered actions and landmarks into a sentence using words with similar meanings and more appropriate words, making them smooth, fluent, and accurate. If the landmarks of adjacent actions are similar or even identical, please use pronouns to refer to them.

  **User**: Multiple sub-instructions.

  **GPT 4o**: Move forward to a high-rise building with a noticeable logo at the top. Then, slightly turn left and go straight to a futuristic tower with a large spherical structure in the middle.

## G Data Quality Control

**Data Filter.** During data collection, it is inevitable that some damaged or low-quality data will be generated. We clean the data in the following situations. 1) We remove damaged images that are produced in generation or transmission. 2) We find that UAVs sometimes appear to pass through the tree models. Therefore, we exclude the trajectories where the altitude is lower than that of the trees. 3) We believe that extremely short or long trajectories are not conducive to model training. Thus, we remove these trajectories, specifically those with fewer than 2 or more than 150 actions.

**Instruction Refinement.** A known challenge of instruction generation is VLMs' hallucinations. During the previous instruction generation process, sometimes the same landmark appears across several frames. This results in a VLM generating similar captions for the repeated observations

of a landmark, increasing the complexity of the final instruction and introducing ambiguity due to duplication.

To mitigate this challenge, we utilize the NLTK library (Bird, 2006) to simplify the instruction by detecting and merging similar descriptions. Specifically, a syntactic parse tree is first constructed to extract all landmark captions using a rule-based approach. Then, a sentence-transformer model is employed to encode the extracted landmark captions into embedding vectors. Their similarities are computed with dot product, and high-similarity captions are then identified and replaced with referential pronouns (*e.g.*, "it," "there," *etc.*). For example, a generated instruction with redundant information is "$\cdots$ make a left turn toward **a medium-sized beige building marked by a signboard reading CHARLIE'S CHOCOLATE**. Continue heading straight, passing **a medium-sized gray building with a prominent rooftop billboard displaying Charlie's Chocolate** $\cdots$". After simplification, a more concise sentence is obtained, *i.e.,* "$\cdots$ make a left turn toward **a medium-sized beige building marked by a signboard reading CHARLIE'S CHOCOLATE**. Continue heading straight, passing **it** $\cdots$", demonstrating the effectiveness of this post-processing technique.

**Manually Check.** At the same time, we built a data inspection platform to provide instructions, action sequences, and corresponding images to human examiners. If an instruction describes all the actions and landmarks in a trajectory well, it is considered qualified. We randomly select 3K samples from the entire dataset according to the data distribution. After manually inspecting these samples, we find that they reach a high qualification rate of 91%. There is some ambiguity in the description of some landmarks in the remaining data, making it likely that these landmarks are not easily distinguishable from the surrounding environment. However, the examiners consider this not entirely unacceptable. In summary, most of the generated data feature good quality for the aerial VLN task.

## H More Dataset Analyses

Following previous studies (Liu et al., 2023; Fan et al., 2022), we conducted a statistical analysis of the linguistic phenomena using 25 randomly selected instructions and compared the results with other VLN datasets, as detailed in Table 5. The analysis shows that the generated instructions exhibit rich linguistic phenomena such as 'Reference' and 'Comparison'. Notably, our dataset is not the most complex one, since we believe that instructions in VLN tasks should be more aligned with real-life scenarios, rather than emphasizing length and complexity. This cognition is consistent with that of REVERIE (Qi et al., 2020). Our instructions avoid overly lengthy and unrealistic expressions to some extent, making them more practical to command UAVs.

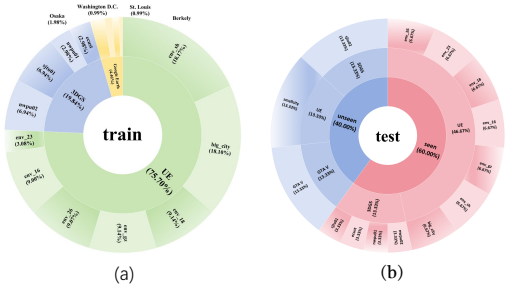



Figure 9: The distribution of the data volume in different scenes under the Train and Test sets. (a) Train set distribution. (b) Test set distribution.

Fig. 9 (a) shows the data distribution of the train set, where 7 UE scenes account for 75.7% of the total 100K data, 4 3D GS scenes account for nearly 20% of the total amount, and Google Earth data accounts for 4.46%. Fig. 9 (b) presents the data distribution of the test set, where the seen data and unseen data account for 60% and 40%, respectively.

## I Qualitative Experimental Results

Fig. 10 presents a qualitative result in a UE scene, where our OpenFly-Agent successfully navigates to the destination according to the instruction. It presents a powerful capability in perceiving environments and aligning observations with complex instructions. Fig. 11 presents another successful aerial VLN example in a 3D GS scene. The image style, flight heights, and viewpoints are significantly different from UE's scenarios. In this case, our OpenFly-Agent exhibits robustness to handle data with great diversity. In addition, Fig. 12 shows two failure cases, where our model sometimes fails to identify the landmark or output actions with proper amplitudes. To further verify the supe-

Table 5: Linguistic phenomena analysis using randomly selected 25 instructions. $p$ denotes the proportion of instructions showing the phenomenon, and $\mu$ represents the average number of times the phenomenon occurs in each instruction.

| Phenomenon | R2R | | ANDH | | AerialVLN | | OpenFly | | Example in OpenFly |
|---|---|---|---|---|---|---|---|---|---|
| | $p$ | $\mu$ | $p$ | $\mu$ | $p$ | $\mu$ | $p$ | $\mu$ | |
| Reference | 100 | 3.7 | 92 | 1.9 | 100 | 9.7 | 100 | 2.7 | ...Advance to **the large beige and brown building with windows**... |
| Coreference | 32 | 0.5 | 8 | 0.1 | 68 | 1.8 | 52 | 1.4 | ...Continue moving forward to reach **it** ... |
| Comparison | 4 | 0.0 | 32 | 0.4 | 20 | 0.2 | 60 | 0.7 | ...Move ahead to the **medium-sized** beige building... |
| Sequencing | 16 | 0.2 | 8 | 0.1 | 68 | 3.7 | 64 | 0.8 | ...move forward to **next** large light brown building ... |
| Allocentric Relation | 20 | 0.2 | 32 | 0.4 | 56 | 4.6 | 76 | 1.0 | ...black large building featuring a billboard **on** its rooftop... |
| Egocentric Relation | 80 | 1.2 | 32 | 0.4 | 100 | 7.1 | 100 | 2.4 | ... then slightly turn left as you move ahead towards... |
| Imperative | 100 | 4.0 | 100 | 1.1 | 100 | 6.9 | 100 | 3.6 | .... Proceed slightly straight and turn left .... |
| Direction | 100 | 2.8 | 100 | 1.4 | 100 | 4.6 | 100 | 3.8 | ... **turn right and go ahead** to ... |
| Temporal Condition | 28 | 0.4 | 20 | 0.2 | 76 | 5.6 | 72 | 0.9 | ...Continue straight **until** you reach ... |

riority of the 'VTM' module, we illustrate the attention maps with and without 'VTM' in Fig. 13, demonstrating its effect in avoiding diluting the attention of the current observation.

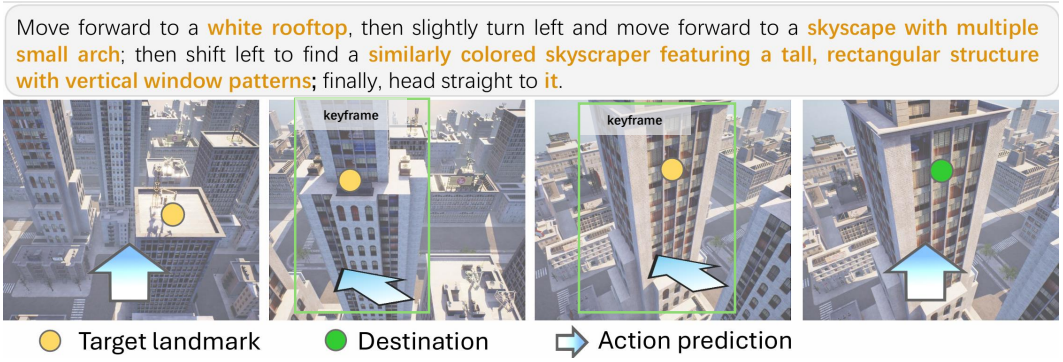



Figure 10: Illustration of an aerial VLN trajectory generated by OpenFly-Agent in a UE scene, which successfully predicts actions following the instruction when encountering landmarks. The green bounding box represents the correct landmark prediction.


Figure 11: Illustration of aerial VLN trajectories generated by OpenFly-Agent in a 3D GS scene. The green bounding box represents the correct landmark locations prediction.

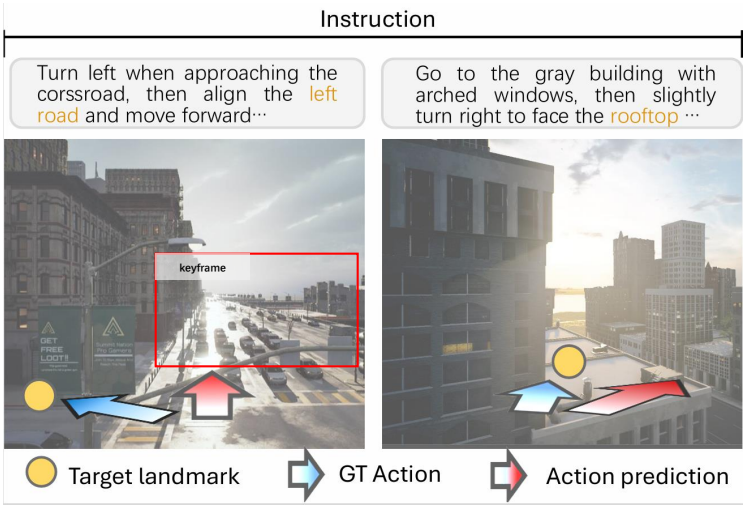



Figure 12: Illustration of failure cases. Sometimes our model may misclassify key landmarks or output wrong actions. The red bounding box represents incorrect landmark locations.

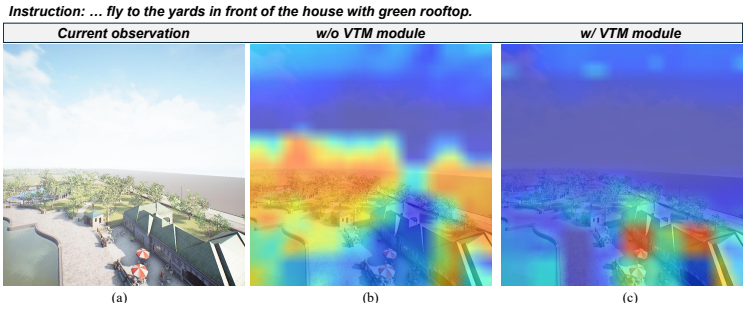



Figure 13: Visualization of attention maps of current observation patches. a) Current observation. b) The attention map without the VTM module and c) the attention map with the VTM module, reflecting token merging strategy matters in avoiding diluting attention of current observation.

## J USE OF LLMS

In this work, we employ large language models (LLMs) to automatically identify and correct grammatical errors, thereby improving the overall fluency and readability of the generated text.